H 5

```
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEEEEEEEEEEEEEE  MMM         MMM  UUU         UUU  LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMMMMM   MMMMMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM   MMM   MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEE     MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAAAAAAAAAAAAAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEE              MMM         MMM  UUU         UUU  LLL              AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
EEEEEEEEEEEEEEE  MMM         MMM  UUUUUUUUUUUUUUU  LLLLLLLLLLLLLLL  AAA         AAA        TTT
```

```
_$2



Sym
---
CMP
DEC
DEC
DEC
DEC

DEC
EXE
EXE
EXE
MMG
MMG
MOV
PRS
SYS
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
VAX
```

```
**FILE**ID**VAXEMULAT                              H 11




VV      VV     AAAAAA     XX      XX   EEEEEEEEEE   MM      MM   UU      UU   LL             AAAAAA     TTTTTTTTTT
VV      VV     AAAAAA     XX      XX   EEEEEEEEEE   MM      MM   UU      UU   LL             AAAAAA     TTTTTTTTTT
VV      VV   AA      AA   XX      XX   EE           MMMM  MMMM   UU      UU   LL           AA      AA       TT
VV      VV   AA      AA   XX      XX   EE           MMMM  MMMM   UU      UU   LL           AA      AA       TT
VV      VV   AA      AA     XX  XX     EE           MM  MM  MM   UU      UU   LL           AA      AA       TT
VV      VV   AA      AA     XX  XX     EE           MM  MM  MM   UU      UU   LL           AA      AA       TT
VV      VV   AA      AA       XX       EEEEEEEE     MM      MM   UU      UU   LL           AA      AA       TT
VV      VV   AA      AA       XX       EEEEEEEE     MM      MM   UU      UU   LL           AA      AA       TT
VV      VV   AAAAAAAAAA     XX  XX     EE           MM      MM   UU      UU   LL           AAAAAAAAAA       TT
VV      VV   AAAAAAAAAA     XX  XX     EE           MM      MM   UU      UU   LL           AAAAAAAAAA       TT
  VV  VV     AA      AA   XX      XX   EE           MM      MM   UU      UU   LL           AA      AA       TT          ....
  VV  VV     AA      AA   XX      XX   EE           MM      MM   UU      UU   LL           AA      AA       TT          ....
    VV       AA      AA   XX      XX   EEEEEEEEEE   MM      MM   UUUUUUUUUU   LLLLLLLLLL   AA      AA       TT          ....
    VV       AA      AA   XX      XX   EEEEEEEEEE   MM      MM   UUUUUUUUUU   LLLLLLLLLL   AA      AA       TT          ....


LL             IIIIII     SSSSSSSS
LL             IIIIII     SSSSSSSS
LL               II       SS
LL               II       SS
LL               II       SS
LL               II       SS
LL               II         SSSSSS
LL               II         SSSSSS
LL               II               SS
LL               II               SS
LL               II               SS
LL               II               SS
LLLLLLLLLL     IIIIII     SSSSSSSS
LLLLLLLLLL     IIIIII     SSSSSSSS
```

Table of contents

```
0000          .NOSHOW CONDITIONALS
0000          .TITLE  VAX$EMULATE - VAX-11 Instruction Emulator
0000          .IDENT  /V04-000/
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility:
0000    ;
0000    ;       VAX-11 Instruction Emulator
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This is the main body of the instruction emulator that supports
0000    ;       the instructions that are not a part of the microVAX architecture.
0000    ;       The current design calls for support of the string instructions
0000    ;       (including CRC), the decimal instructions, and EDITPC.
0000    ;
0000    ;       This routine performs the following steps.
0000    ;
0000    ;       o Moves operands from the exception stack to registers in an
0000    ;         instruction-specific manner
0000    ;
0000    ;       o Calls an instruction-specific subroutine to do the actual work
0000    ;
0000    ;       If errors occur along the way, those errors are reflected to the
0000    ;       user as exceptions.
0000    ;
0000    ; Environment:
0000    ;
0000    ;       These routines run at any access mode, at any IPL, and are AST
0000    ;       reentrant. The routine starts execution in the access mode and
0000    ;       at the IPL at which the instruction executed.
0000    ;
```

```
0000   ; Author:
0000   ;
0000   ;       Lawrence J. Kenah
0000   ;
0000   ; Creation Date
0000   ;
0000   ;       17 August 1982
0000   ;
0000   ; Modified by:
0000   ;
0000   ;       V01-011 LJK0041         Lawrence J. Kenah       16-Jul-1984
0000   ;               Clear FPD in saved PSL at VAX$EMULATE_FPD entry so that
0000   ;               next instruction can execute correctly.
0000   ;
0000   ;       V01-010 LJK0031         Lawrence J. Kenah       5-Jul-1984
0000   ;               Set R2 and R4 unconditionally to zero in EDITPC routine
0000   ;               to allow the storage of FPD flags and similar data.
0000   ;
0000   ;       V01-009 LJK0026         Lawrence J. Kenah       19-Mar-1984
0000   ;               Perform final cleanup pass. Eliminate xxx_UNPACK routine
0000   ;               references. Add C-bit optimization to MOVP.
0000   ;
0000   ;       V01-008 LJK0010         Lawrence J. Kenah       8-Nov-1983
0000   ;               Eliminate code in EXIT_EMULATOR path that unconditionally
0000   ;               clears the T-bit and conditionally sets the TP-bit. The
0000   ;               TP-bit is handled by the base hardware.
0000   ;
0000   ;       V01-007 KDM0088         Kathleen D. Morse       20-Oct-1983
0000   ;               Make branches to VAX$REFLECT_TO_VMS into jumps, so that
0000   ;               the bootstrap emulator will link without truncation errors
0000   ;               until that routine is finished.
0000   ;
0000   ;       V01-006 KDM0003         Kathleen D. Morse       18-Apr-1983
0000   ;               Generate abbreviated VAX$EMULATE_FPD for the bootstrap
0000   ;               emulator.
0000   ;
0000   ;       V01-005 LJK0006         Lawrence J. Kenah       16-Mar-1983
0000   ;               Generate case tables with macros. Allow subset emulator
0000   ;               for bootstrap instruction emulation.
0000   ;
0000   ;       V01-004 KDM0002         Kathleen D. Morse       16-Mar-1983
0000   ;               Fix fourth and fifth operand fetches for SUBP6, ADDP6,
0000   ;               MULP and DIVP.
0000   ;
0000   ;       V01-003 KDM0001         Kathleen D. Morse       04-Mar-1983
0000   ;               Longword align the exception handler entry points.
0000   ;
0000   ;       V01-002 LJK0005         Lawrence J. Kenah       15-Nov-1982
0000   ;               Use hardware aids provided by microVAX architecture revision.
0000   ;               Exception is now reported in caller's mode. Operands are parsed
0000   ;               and placed on the exception stack as exception parameters.
0000   ;
0000   ;       V01-001 LJK0002         Lawrence J. Kenah       17-Aug-1982
0000   ;               Original version using kernel mode exception through OPCDEC
0000   ;               exception vector.
0000   ;
0000   ;--
```

```
0000            .SUBTITLE       DECLARATIONS
0000
0000    ; Include files:
0000
0000            $OPDEF                                  ; Values for instruction opcodes
0000            $PSLDEF                                 ; Define bit fields in PSL
0000
0000            .NOCROSS                                ; No cross reference for these
0000            .ENABLE         SUPPRESSION             ; No symbol table entries either
0000
0000            PACK_DEF                                ; Stack usage when restarting instructions
0000            STACK_DEF                               ; Stack usage for original exception
0000
0000            .DISABLE        SUPPRESSION             ; Turn on symbol table again
0000            .CROSS                                  ; Cross reference is OK now
0000
0000    ; Macro definitions
0000
0000            .MACRO  INIT_CASE_TABLE         SIZE,BASE,ERROR_EXIT
0000    BASE:
0000            .REPT   SIZE
0000            .WORD   ERROR_EXIT-BASE
0000            .ENDR
0000            .ENDM   INIT_CASE_TABLE
0000
0000            .MACRO  CASE_TABLE_ENTRY        OPCODE,-
0000                                            ROUTINE,-
0000                                            FPD_ROUTINE,-
0000                                            BOOT_FLAG
0000                    SIGN_EXTEND     OP$_'OPCODE , ...OPCODE
0000                    ...OFFSET = ...OPCODE - OPCODE_BASE
0000                    .IF     NOT_DEFINED     BOOT_SWITCH
0000                            INCLUDE_'OPCODE = 0
0000                            .EXTERNAL       VAX$'OPCODE
0000                            .EXTERNAL       FPD_ROUTINE
0000                            . = CASE_TABLE_BASE + <2 * ...OFFSET>
0000                            .WORD   ROUTINE - CASE_TABLE_BASE
0000                            . = FPD_CASE_TABLE_BASE + <2 * ...OFFSET>
0000                            .WORD   FPD_ROUTINE - FPD_CASE_TABLE_BASE
0000                    .IF_FALSE
0000                            .IF     IDENTICAL       <BOOT_FLAG>,BOOT
0000                            INCLUDE_'OPCODE = 0
0000                            .EXTERNAL       VAX$'OPCODE
0000                            . = CASE_TABLE_BASE + <2 * ...OFFSET>
0000                            .WORD   ROUTINE - CASE_TABLE_BASE
0000                            .ENDC
0000                    .ENDC
0000                    .ENDM   CASE_TABLE_ENTRY
0000
0000    ; External declarations for exception handling
0000
0000            .DISABLE        GLOBAL
0000
0000            .EXTERNAL       VAX$AL_DELTA_PC_TABLE
0000            .EXTERNAL       VAX$REFLECT_TO_VMS
0000
0000            .EXTERNAL       VAX$_OPCDEC, -
```

```
    0000   178                          VAX$_OPCDEC_FPD
    0000   179
    0000   180 ; PSECT Declarations:
    0000   181
    0000   182          .DEFAULT        DISPLACEMENT , WORD
    0000   183
00000000   184          .PSECT _VAX$CODE PIC, USR, CON, REL, LCL, SHR, EXE, RD, NOWRT, QUAD
    0000   185
    0000   186          .NOSHOW         CONDITIONALS
```

```
0000          .SUBTITLE       VAX$EMULATE - Entry Path into Emulator
0000  ;+
0000  ; Functional Description:
0000  ;
0000  ;       There are two different entries into this module. When a reserved
0000  ;       instruction is first encountered, its operands are parsed by the
0000  ;       hardware (or microcode, if you will) and placed on the stack as
0000  ;       exception parameters. The code at address VAX$EMULATE is then entered
0000  ;       through the ^XC8(SCB) exception vector. That routine dispatches to an
0000  ;       instruction-specific routine called VAX$xxxxxx (xxxxxx represents the
0000  ;       name of the reserved instruction) after placing the operands into
0000  ;       registers as required by VAX$xxxxxx.
0000  ;
0000  ;       If an exception occurred during instruction emulation such that a
0000  ;       reserved instruction executed again, this time with FPD set, then a
0000  ;       different exception path is taken. The stack has a different (smaller)
0000  ;       set of parameters for the FPD exception. A different
0000  ;       instruction-specific routine executes to unpack saved intermediate
0000  ;       state before resuming instruction emulation.
0000  ;
0000  ;       The access mode and IPL are preserved across either exception.
0000  ;
0000  ; Input Parameters:
0000  ;
0000  ;       00(SP) - Opcode of reserved instruction
0000  ;       04(SP) - PC of reserved instruction (old PC)
0000  ;       08(SP) - First operand specifier
0000  ;       12(SP) - Second operand specifier
0000  ;       16(SP) - Third operand specifier
0000  ;       20(SP) - Fourth operand specifier
0000  ;       24(SP) - Fifth operand specifier
0000  ;       28(SP) - Sixth operand specifier
0000  ;       32(SP) - Seventh operand specifier (currently unused)
0000  ;       36(SP) - Eight operand specifier (currently unused)
0000  ;       40(SP) - PC of instruction following reserved instruction (new PC)
0000  ;       44(SP) - PSL at time of exception
0000  ;
0000  ; Notes on input parameters:
0000  ;
0000  ;   1.  The information that appears on the stack for each operand depends
0000  ;       on the nature of the operand.
0000  ;
0000  ;       .rx - Operand value
0000  ;       .ax - Operand address
0000  ;       .wx - Operand address (Register destination is stored in one's
0000  ;             complement form. See VAX$CVTPL for details.)
0000  ;
0000  ;   2.  The old PC value is not used unless an exception such as an access
0000  ;       violation occurs and the instruction has to be backed up.
0000  ;
0000  ;   3.  The seventh and eighth operands are not used for any existing VAX-11
0000  ;       instructions. Those slots in the exception stach frame are reserved
0000  ;       for future expansion.
0000  ;
0000  ;   4.  The two PC parameters and the PSL are the only data that needs to
0000  ;       be preserved once the instruction-specific routine is entered.
0000  ;
```

```
                                 0000   ; Output Parameters:
                                 0000   ;
                                 0000   ;       The operands are moved from the stack to general registers in a way
                                 0000   ;       that varies from instruction to instruction. Control is transferred
                                 0000   ;       to a specific routine for each opcode.
                                 0000   ;
                                 0000   ; Notes:
                                 0000   ;
                                 0000   ;       There are several tables in the emulator that use the opcode as an
                                 0000   ;       index. We choose to interpret the opcode as a signed quantity because
                                 0000   ;       this reduces the amount of wasted space in the tables. In either case,
                                 0000   ;       there are 27 useful entries.
                                 0000   ;
                                 0000   ;       Unsigned opcode
                                 0000   ;
                                 0000   ;               OPCODE_BASE = CVTPS (value of 8)
                                 0000   ;               OPCODE_MAX = CVTLP (value of F9)
                                 0000   ;
                                 0000   ;               TABLE_SIZE = 241 decimal bytes
                                 0000   ;
                                 0000   ;       Signed opcode
                                 0000   ;
                                 0000   ;               OPCODE_BASE = ASHP (value of F8 or -8)
                                 0000   ;               OPCODE_MAX = SKPC (value of 3B)
                                 0000   ;
                                 0000   ;               TABLE_SIZE = 67 decimal bytes
                                 0000   ;
                                 0000   ;       The savings of more than 170 entries in each table justifies all
                                 0000   ;       of the machinations that we go through to treat opcodes as signed
                                 0000   ;       quantities.
                                 0000   ;-
                                 0000
                                 0000   ; Because the assembler does not understand sign extension of byte and
                                 0000   ; word quantities, we must accomplish this sign extension with macros. The
                                 0000   ; assignment statements that appear as comments illustrate the sense of the
                                 0000   ; macro invocations that immediately follow.
                                 0000
                                 0000   ;       OPCODE_MAX = OP$_SKPC           ; Largest opcode in this emulator
                                 0000
                                 0000           SIGN_EXTEND     OP$_SKPC , OPCODE_MAX
                                 0000
                                 0000   ; We further restrict the table size and supported operations when we are
                                 0000   ; building the bootstrap subset of the emulator. We only allow certain string
                                 0000   ; instructions to contribute to the emulator.
                                 0000
                                 0000   ;       OPCODE_BASE = OP$_ASHP          ; Smallest (in signed sense) opcode
                                 0000
                                 0000           SIGN_EXTEND     OP$_ASHP , OPCODE_BASE
                                 0000
                     00000044    0000   CASE_TABLE_SIZE = <OPCODE_MAX - OPCODE_BASE> + 1        ; Define table size
                                 0000
                                 0000           .ALIGN  LONG                    ; Alignment for exception vector
                                 0000
                                 0000   VAX$EMULATE::
                                 0000
        43 8F   F8 8F   6E   8F  0000           CASEB   OPCODE(SP),#OPCODE_BASE,#<OPCODE_MAX-OPCODE_BASE>
                                 0006
```

```
                           0006          INIT_CASE_TABLE CASE_TABLE_SIZE,CASE_TABLE_BASE,10$
                           008E
                           008E   ; If we drop through the case dispatcher, then the fault was not caused
                           008E   ; by executing one of the instructions supported by this emulator. Such
                           008E   ; exceptions will simply be passed through to VMS. (In the bootstrap emulator,
                           008E   ; there is no operating system to reflect the exception. We simply HALT.)
                           008E
         00000000'8F   DD  008E   10$:    PUSHL   #VAX$_OPCDEC             ; Store signal name
                  0D   DD  0094           PUSHL   #13                      ; Total of 13 longwords in signal array
                           0096
               FF67'   31  0096           BRW     VAX$REFLECT_TO_VMS       ; Use common exit to VMS
```

```
0099            .SUBTITLE       VAX$EMULATE_FPD - Alternate Entry Path into Emulator
0099    ;+
0099    ; Functional Description:
0099    ;
0099    ;       This routine is entered through the ^XCC(SCB) exception vector when an
0099    ;       instruction that is not a part of the microVAX architecture executes
0099    ;       and the FPD bit is set in the PSL. The software state that was
0099    ;       preserved by each instruction must be restored and instruction
0099    ;       execution resumed. Access mode and IPL are preserved across the
0099    ;       exception occurrence.
0099    ;
0099    ;       Before the various VAX$xxxxxx (or VAX$xxxxxx_RESTART) routines regain
0099    ;       control, this dispatcher must retrieve the delta PC from wherever
0099    ;       it was stored and place the stack in the same state that it is in
0099    ;       when the normal (FPD bit not set) instruction dispatcher passes
0099    ;       control to the various VAX$xxxxxx routines. The pictures below explain
0099    ;       this.
0099    ;
0099    ; Input Parameters:
0099    ;
0099    ;       00(SP) - PC of reserved instruction
0099    ;       04(SP) - PSL at time of exception
0099    ;
0099    ; Output Parameters:
0099    ;
0099    ;       The following picture shows the state of the stack after the dispatcher
0099    ;       has executed its preliminary code but before control is passed back to
0099    ;       instruction-specific  execution. Note that this routine makes the
0099    ;       stack look like it does when a reserved instruction executes and FPD
0099    ;       is not yet set. This is done to make the exception exit code independent
0099    ;       of whether a different exception exception occurred while the emulator
0099    ;       was running.
0099    ;
0099    ;       00(SP) - Return PC (Address of EXIT routine in this module)
0099    ;       04(SP) - Unused placeholder (OPCODE)
0099    ;       08(SP) - PC of reserved instruction (old PC)
0099    ;       12(SP) - Unused placeholder (OPERAND_1)
0099    ;       16(SP) - Unused placeholder (OPERAND_2)
0099    ;       20(SP) - Unused placeholder (OPERAND_3)
0099    ;       24(SP) - Unused placeholder (OPERAND_4)
0099    ;       28(SP) - Unused placeholder (OPERAND_5)
0099    ;       32(SP) - Unused placeholder (OPERAND_6)
0099    ;       36(SP) - Unused placeholder (OPERAND_7)
0099    ;       40(SP) - Unused placeholder (OPERAND_8)
0099    ;       44(SP) - PC of instruction following reserved instruction (new PC)
0099    ;       48(SP) - PSL at time of exception
0099    ;
0099    ;       Before this routine dispatches to opcode-specific code, it calculates
0099    ;       the PC of the next instruction based on the PC of the reserved
0099    ;       instruction and the delta-PC quantity that was stored as part of the
0099    ;       instruction's intermediate state. Note that the delta PC quantity
0099    ;
0099    ;               delta PC = new PC - old PC
0099    ;
0099    ;       is stored in the upper bytes of one of the general registers, usually
0099    ;       bits <31:24> of R0 or R2. The registers R0 through R3 are stored on
0099    ;       the stack (in the space used for the first four operands when the
                                   0099  ;       reserved instruction is first encountered) so that the same offsets
                                   0099  ;       that were used to store the delta-PC can be used to retrieve it.
                                   0099  ;-
                                   0099
                                   0099          .ALIGN  LONG                    ; Alignment for exception vector
                                   009C
                                   009C  VAX$EMULATE_FPD::
                                   009C
                                   009C
               00 04 AE   1B   E5  009C          BBCC    #PSL$V_FPD,4(SP),5$     ; Clear FPD in exception PSL
                     5E   28   C2  00A1  5$:     SUBL2   #NEW_PC,SP              ; Create extra stack space
               04 AE   28 AE   D0  00A4          MOVL    NEW_PC(SP),OLD_PC(SP)   ; Make second copy of old PC
                  08 AE   50   7D  00A9          MOVQ    R0,OPERAND_1(SP)        ; Save R0 and R1 in some extra space
                  10 AE   52   7D  00AD          MOVQ    R2,OPERAND_3(SP)        ; Do the same for R2 and R3
                  50   04 BE   98  00B1          CVTBL   @OLD_PC(SP),R0          ; Get opcode from instruction stream
              51    0000'CF40  9A  00B5          MOVZBL  VAX$AL_DELTA_PC_TABLE[R0],R1 ; Get offset to byte with delta-PC
                51    08 AE41  9A  00BB          MOVZBL  OPERAND_1(SP)[R1],R1    ; Get delta-PC
                  28 AE   51   C0  00C0          ADDL    R1,NEW_PC(SP)           ; Convert old PC to new PC
                     6E   50   D0  00C4          MOVL    R0,OPCODE(SP)           ; Store opcode in other than a register
                  50   08 AE   7D  00C7          MOVQ    OPERAND_1(SP),R0        ; Restore R0 and R1
                                   00CB                                          ; (R2 and R3 were not changed)
                     0425'CF   9F  00CB          PUSHAB  VAX$EXIT_EMULATOR       ; Create return PC to make CASE like BSB
                                   00CF
        43 8F   F8 8F   04 AE  8F  00CF          CASEB   <OPCODE+4>(SP),#OPCODE_BASE,#<OPCODE_MAX-OPCODE_BASE>
                                   00D6
                                   00D6          INIT_CASE_TABLE CASE_TABLE_SIZE,FPD_CASE_TABLE_BASE,10$
                                   015E
                                   015E  ; If we drop through the case dispatcher, then the fault was not caused
                                   015E  ; by executing one of the instructions supported by this emulator. The
                                   015E  ; exception will be passed to VMS with the following stack.
                                   015E  ;
                                   015E  ;       00(SP) - Signal array size (always 4)
                                   015E  ;       04(SP) - Signal name (VAX$_OPCDEC_FPD)
                                   015E  ;       08(SP) - Opcode that is not supported
                                   015E  ;       12(SP) - PC of that opcode
                                   015E  ;       16(SP) - PSL of exception
                                   015E  ;
                                   015E  ; (In the bootstrap emulator, we simply halt with the stack containing
                                   015E  ; these data.)
                                   015E
                     5E   04   C0  015E  10$:    ADDL    #4,SP                   ; Discard return PC
               28 AE   04 AE   D0  0161          MOVL    OLD_PC(SP),NEW_PC(SP)   ; Use PC of opcode and not new PC
                  24 AE   6E   D0  0166          MOVL    OPCODE(SP),OPERAND_8(SP)  ; Include opcode in signal array
                  5E   24 AE   DE  016A          MOVAL   OPERAND_8(SP),SP        ; Discard rest of stack
                                   016E
                                   016E
                 00000000'8F   DD  016E          PUSHL   #VAX$_OPCDEC_FPD        ; This is the signal name
                          04   DD  0174          PUSHL   #4                      ; Signal array has four longwords
                                   0176
                       FE87'   31  0176          BRW     VAX$REFLECT_TO_VMS      ; Use common exit to VMS
```

```
0179                 .SUBTITLE       Dispatch Tables
0179         ;+
0179         ; Functional Description:
0179         ;
0179         ;       The case tables for the two CASEB instructions are built with the
0179         ;       macros that are invoked here. Macros are used to guarantee that both
0179         ;       tables contain correct entries for a selected opcode at the same
0179         ;       offset.
0179         ;
0179         ; Assumptions:
0179         ;
0179         ;       The CASE_TABLE_ENTRY macro assumes that the names of the respective
0179         ;       case tables are CASE_TABLE_BASE and FPD_CASE_TABLE_BASE.
0179         ;
0179         ; Notes:
0179         ;
0179         ;       In the following lists, those FPD routines that do not have _FPD in
0179         ;       their names use the same JSB entry point for initial entry and after
0179         ;       restarting the instruction. In most of these cases, the register state
0179         ;       is the same for both starting and restarting. For the remaining cases,
0179         ;       there is not enough difference between the two cases to justify an
0179         ;       additional entry point. (See VAX$MOVTC for an example of this latter
0179         ;       situation.)
0179         ;
0179         ;       The FPD routines that include _RESTART in their names have to do a
0179         ;       certain amount of work to restore the intermediate state from the
0179         ;       canonical registers before they can resume instruction execution.
0179         ;-
0179
0179                 .SAVE                                   ; Remember current location counter
0179
0179         ; First generate table entries for the string instructions
0179
0179                 CASE_TABLE_ENTRY        OPCODE=MOVTC,-
0179                                         ROUTINE=MOVTC,-
0179                                         FPD_ROUTINE=VAX$MOVTC
0144
0144                 CASE_TABLE_ENTRY        OPCODE=MOVTUC,-
0144                                         ROUTINE=MOVTUC,-
0144                                         FPD_ROUTINE=VAX$MOVTUC
0146
0146                 CASE_TABLE_ENTRY        OPCODE=CMPC3,-
0146                                         ROUTINE=CMPC3,-
0146                                         FPD_ROUTINE=VAX$CMPC3,-
0146                                         BOOT_FLAG=BOOT
013A
013A                 CASE_TABLE_ENTRY        OPCODE=CMPC5,-
013A                                         ROUTINE=CMPC5,-
013A                                         FPD_ROUTINE=VAX$CMPC5,-
013A                                         BOOT_FLAG=BOOT
0142
0142                 CASE_TABLE_ENTRY        OPCODE=LOCC,-
0142                                         ROUTINE=LOCC,-
0142                                         FPD_ROUTINE=VAX$LOCC,-
0142                                         BOOT_FLAG=BOOT
015C
015C                 CASE_TABLE_ENTRY        OPCODE=SKPC,-
```

```
0150                                      ROUTINE=SKPC,-
0150                                      FPD_ROUTINE=VAX$SKPC
015E
015E         CASE_TABLE_ENTRY             OPCODE=SCANC,-
015E                                      ROUTINE=SCANC,-
015E                                      FPD_ROUTINE=VAX$SCANC
013C
013C         CASE_TABLE_ENTRY             OPCODE=SPANC,-
013C                                      ROUTINE=SPANC,-
013C                                      FPD_ROUTINE=VAX$SPANC
013E
013E         CASE_TABLE_ENTRY             OPCODE=MATCHC,-
013E                                      ROUTINE=MATCHC,-
013E                                      FPD_ROUTINE=VAX$MATCHC
015A
015A         CASE_TABLE_ENTRY             OPCODE=CRC,-
015A                                      ROUTINE=CRC,-
015A                                      FPD_ROUTINE=VAX$CRC
00FE
00FE  ; Now generate table entries for the decimal instructions
00FE
00FE         CASE_TABLE_ENTRY             OPCODE=ADDP4,-
00FE                                      ROUTINE=ADDP4,-
00FE                                      FPD_ROUTINE=VAX$ADDP4
0128
0128         CASE_TABLE_ENTRY             OPCODE=ADDP6,-
0128                                      ROUTINE=ADDP6,-
0128                                      FPD_ROUTINE=VAX$ADDP6
012A
012A         CASE_TABLE_ENTRY             OPCODE=ASHP,-
012A                                      ROUTINE=ASHP,-
012A                                      FPD_ROUTINE=VAX$ASHP
00D8
00D8         CASE_TABLE_ENTRY             OPCODE=CMPP3,-
00D8                                      ROUTINE=CMPP3,-
00D8                                      FPD_ROUTINE=VAX$CMPP3
0152
0152         CASE_TABLE_ENTRY             OPCODE=CMPP4,-
0152                                      ROUTINE=CMPP4,-
0152                                      FPD_ROUTINE=VAX$CMPP4
0156
0156         CASE_TABLE_ENTRY             OPCODE=CVTLP,-
0156                                      ROUTINE=CVTLP,-
0156                                      FPD_ROUTINE=VAX$CVTLP_RESTART
00DA
00DA         CASE_TABLE_ENTRY             OPCODE=CVTPL,-
00DA                                      ROUTINE=CVTPL,-
00DA                                      FPD_ROUTINE=VAX$CVTPL_RESTART
0154
0154         CASE_TABLE_ENTRY             OPCODE=CVTPS,-
0154                                      ROUTINE=CVTPS,-
0154                                      FPD_ROUTINE=VAX$CVTPS
00F8
00F8         CASE_TABLE_ENTRY             OPCODE=CVTPT,-
00F8                                      ROUTINE=CVTPT,-
00F8                                      FPD_ROUTINE=VAX$CVTPT_RESTART
0130
```

```
    0130          CASE_TABLE_ENTRY        OPCODE=CVTSP,-
    0130                                  ROUTINE=CVTSP,-
    0130                                  FPD_ROUTINE=VAX$CVTSP
    00FA
    00FA          CASE_TABLE_ENTRY        OPCODE=CVTTP,-
    00FA                                  ROUTINE=CVTTP,-
    00FA                                  FPD_ROUTINE=VAX$CVTTP_RESTART
    0134
    0134          CASE_TABLE_ENTRY        OPCODE=DIVP,-
    0134                                  ROUTINE=DIVP,-
    0134                                  FPD_ROUTINE=VAX$DIVP
    0136
    0136          CASE_TABLE_ENTRY        OPCODE=MOVP,-
    0136                                  ROUTINE=MOVP,-
    0136                                  FPD_ROUTINE=VAX$MOVP
    0150
    0150          CASE_TABLE_ENTRY        OPCODE=MULP,-
    0150                                  ROUTINE=MULP,-
    0150                                  FPD_ROUTINE=VAX$MULP
    0132
    0132          CASE_TABLE_ENTRY        OPCODE=SUBP4,-
    0132                                  ROUTINE=SUBP4,-
    0132                                  FPD_ROUTINE=VAX$SUBP4
    012C
    012C          CASE_TABLE_ENTRY        OPCODE=SUBP6,-
    012C                                  ROUTINE=SUBP6,-
    012C                                  FPD_ROUTINE=VAX$SUBP6
    012E
    012E  ; EDITPC always seems to find itself in last place
    012E
    012E          CASE_TABLE_ENTRY        OPCODE=EDITPC,-
    012E                                  ROUTINE=EDITPC,-
    012E                                  FPD_ROUTINE=VAX$EDITPC_RESTART
    0158
00000179          .RESTORE                                ; Reset current location counter
```

```
0179            .SUBTITLE       Description of instruction-specific routines
0179
0179    ;++
0179    ; The instruction-specific routines do similar things. Rather than clutter up
0179    ; each routine with the same comments, we will describe the steps that each
0179    ; routine takes in this section.
0179    ;
0179    ; The input parameters to each routine are identical.
0179    ;
0179    ;                       Contents of exception stack
0179    ;                       ---------------------------
0179    ;
0179    ;       OPCODE(SP)      - Opcode of reserved instruction
0179    ;       OLD_PC(SP)      - PC of reserved instruction
0179    ;       OPERAND_1(SP) - First operand specifier
0179    ;       OPERAND_2(SP) - Second operand specifier
0179    ;       OPERAND_3(SP) - Third operand specifier
0179    ;       OPERAND_4(SP) - Fourth operand specifier
0179    ;       OPERAND_5(SP) - Fifth operand specifier
0179    ;       OPERAND_6(SP) - Sixth operand specifier
0179    ;       OPERAND_7(SP) - Seventh operand specifier (currently unused)
0179    ;       OPERAND_8(SP) - Eight operand specifier (currently unused)
0179    ;       NEW_PC(SP)      - PC of instruction following reserved instruction
0179    ;       EXCEPTION_PSL(SP) - PSL at time of exception
0179    ;
0179    ;       The routine headers for the instruction-specific routines in this
0179    ;       module will list the input and output parameters in symbolic form
0179    ;       only. The VAX$xxxxxx routines in other modules in the emulator contain
0179    ;       the exact meanings of the various operands (parameters) to the
0179    ;       routines.
0179    ;
0179    ; Outline of execution:
0179    ;
0179    ;       The operands are loaded into registers as required by the instruction
0179    ;       specific routines. Routine headers for each routine contain detailed
0179    ;       descriptions.
0179    ;
0179    ;       A routine of the form VAX$xxxxxx (where xxxxxx is the instruction
0179    ;       name) is called to perform the actual work indicated by each
0179    ;       instruction.
0179    ;
0179    ;       Common exit code executes to allow the condition codes returned by the
0179    ;       VAX$xxxxxx routines to be passed back to the code that generated the
0179    ;       original exception.
0179    ;
0179    ; Notes:
0179    ;
0179    ;       The following routines are constructed to be reasonably fast. In
0179    ;       particular, each instruction has its own separate routine, even though
0179    ;       several instructions differ only in the instruction-specific routine
0179    ;       to which final control is passed. Rather than share this common code
0179    ;       at the expense of another dispatch on opcode, we shoose to duplicate
0179    ;       the common code.
0179    ;--
0179
```

```
                          0179          .SUBTITLE       MOVTC - Exception handler for MOVTC instruction
                          0179  ;+
                          0179  ; Input Parameters:
                          0179  ;
                          0179  ;       OPCODE(SP)
                          0179  ;       OLD_PC(SP)
                          0179  ;       OPERAND_1(SP) - srclen.rw
                          0179  ;       OPERAND_2(SP) - srcaddr.ab
                          0179  ;       OPERAND_3(SP) - fill.rb
                          0179  ;       OPERAND_4(SP) - tbladdr.ab
                          0179  ;       OPERAND_5(SP) - dstlen.rw
                          0179  ;       OPERAND_6(SP) - dstaddr.ab
                          0179  ;       OPERAND_7(SP)
                          0179  ;       OPERAND_8(SP)
                          0179  ;       NEW_PC(SP)
                          0179  ;       EXCEPTION_PSL(SP)
                          0179  ;
                          0179  ; Output Parameters:
                          0179  ;
                          0179  ;       R0<15:0> - srclen.rw
                          0179  ;       R1       - srcaddr.ab
                          0179  ;       R2<7:0>  - fill.rb
                          0179  ;       R3       - tbladdr.ab
                          0179  ;       R4<15:0> - dstlen.rw
                          0179  ;       R5       - dstaddr.ab
                          0179  ;
                          0179  ; Implicit Output:
                          0179  ;
                          0179  ;       R0<31:16> - 0
                          0179  ;       R2<31:8>  - 0
                          0179  ;       R4<31:16> - 0
                          0179  ;-
                          0179
                          0179  MOVTC:
                          0179
       50   08 AE    3C   0179          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
       51   0C AE    D0   017D          MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
       52   10 AE    9A   0181          MOVZBL  OPERAND_3(SP),R2        ; R2<7:0>  <- fill.rb
       53   14 AE    D0   0185          MOVL    OPERAND_4(SP),R3        ; R3       <- tbladdr.ab
       54   18 AE    3C   0189          MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- dstlen.rw
       55   1C AE    D0   018D          MOVL    OPERAND_6(SP),R5        ; R5       <- dstaddr.ab
                          0191
                          0191  ; Now that the operands have been loaded, the only exception parameter
                          0191  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          0191  ; there is no reason why the state of the stack needs to be altered and we
                          0191  ; save two instructions if we leave the stack alone.
                          0191
          0425'CF    9F   0191          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
            FE68'    31   0195          BRW     VAX$MOVTC               ; Do the actual work
```

```
                      0198        .SUBTITLE       MOVTUC - Exception handler for MOVTUC instruction
                      0198  ;+
                      0198  ; Input Parameters:
                      0198  ;
                      0198  ;       OPCODE(SP)
                      0198  ;       OLD_PC(SP)
                      0198  ;       OPERAND_1(SP) - srclen.rw
                      0198  ;       OPERAND_2(SP) - srcaddr.ab
                      0198  ;       OPERAND_3(SP) - esc.rb
                      0198  ;       OPERAND_4(SP) - tbladdr.ab
                      0198  ;       OPERAND_5(SP) - dstlen.rw
                      0198  ;       OPERAND_6(SP) - dstaddr.ab
                      0198  ;       OPERAND_7(SP)
                      0198  ;       OPERAND_8(SP)
                      0198  ;       NEW_PC(SP)
                      0198  ;       EXCEPTION_PSL(SP)
                      0198  ;
                      0198  ; Output Parameters:
                      0198  ;
                      0198  ;       R0<15:0> - srclen.rw
                      0198  ;       R1       - srcaddr.ab
                      0198  ;       R2<7:0>  - esc.rb
                      0198  ;       R3       - tbladdr.ab
                      0198  ;       R4<15:0> - dstlen.rw
                      0198  ;       R5       - dstaddr.ab
                      0198  ;
                      0198  ; Implicit Output:
                      0198  ;
                      0198  ;       R0<31:16> - 0
                      0198  ;       R2<31:8>  - 0
                      0198  ;       R4<31:16> - 0
                      0198  ;-
                      0198
                      0198  MOVTUC:
                      0198
50    08 AE    3C     0198        MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
51    0C AE    D0     019C        MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
52    10 AE    9A     01A0        MOVZBL  OPERAND_3(SP),R2        ; R2<7:0>  <- esc.rb
53    14 AE    D0     01A4        MOVL    OPERAND_4(SP),R3        ; R3       <- tbladdr.ab
54    18 AE    3C     01A8        MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- dstlen.rw
55    1C AE    D0     01AC        MOVL    OPERAND_6(SP),R5        ; R5       <- dstaddr.ab
                      01B0
                      01B0  ; Now that the operands have been loaded, the only exception parameter
                      01B0  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                      01B0  ; there is no reason why the state of the stack needs to be altered and we
                      01B0  ; save two instructions if we leave the stack alone.
                      01B0
     0425'CF   9F     01B0        PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
        FE49'  31     01B4        BRW     VAX$MOVTUC              ; Do the actual work
```

```
                       01B7          .SUBTITLE       CMPC3 - Exception handler for CMPC3 instruction
                       01B7  ;+
                       01B7  ; Input Parameters:
                       01B7  ;
                       01B7  ;       OPCODE(SP)
                       01B7  ;       OLD_PC(SP)
                       01B7  ;       OPERAND_1(SP) - len.rw
                       01B7  ;       OPERAND_2(SP) - src1addr.ab
                       01B7  ;       OPERAND_3(SP) - src2addr.ab
                       01B7  ;       OPERAND_4(SP)
                       01B7  ;       OPERAND_5(SP)
                       01B7  ;       OPERAND_6(SP)
                       01B7  ;       OPERAND_7(SP)
                       01B7  ;       OPERAND_8(SP)
                       01B7  ;       NEW_PC(SP)
                       01B7  ;       EXCEPTION_PSL(SP)
                       01B7  ;
                       01B7  ; Output Parameters:
                       01B7  ;
                       01B7  ;       R0<15:0> - len.rw
                       01B7  ;       R1       - src1addr.ab
                       01B7  ;       R3       - src2addr.ab
                       01B7  ;
                       01B7  ; Implicit Output:
                       01B7  ;
                       01B7  ;       R0<31:16> - 0
                       01B7  ;       R2        - UNPREDICTABLE
                       01B7  ;-
                       01B7
                       01B7  CMPC3:
                       01B7
   50   08 AE    3C    01B7          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
   51   0C AE    D0    01BB          MOVL    OPERAND_2(SP),R1        ; R1       <- src1addr.ab
   53   10 AE    D0    01BF          MOVL    OPERAND_3(SP),R3        ; R3       <- src2addr.ab
                       01C3
                       01C3  ; Now that the operands have been loaded, the only exception parameter
                       01C3  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                       01C3  ; there is no reason why the state of the stack needs to be altered and we
                       01C3  ; save two instructions if we leave the stack alone.
                       01C3
      0425'CF    9F    01C3          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
         FE36'   31    01C7          BRW     VAX$CMPC3               ; Do the actual work
```

```
                              01CA          .SUBTITLE       CMPC5 - Exception handler for CMPC5 instruction
                              01CA  ;+
                              01CA  ; Input Parameters:
                              01CA  ;
                              01CA  ;       OPCODE(SP)
                              01CA  ;       OLD_PC(SP)
                              01CA  ;       OPERAND_1(SP) - src1len.rw
                              01CA  ;       OPERAND_2(SP) - src1addr.ab
                              01CA  ;       OPERAND_3(SP) - fill.rb
                              01CA  ;       OPERAND_4(SP) - src2len.rw
                              01CA  ;       OPERAND_5(SP) - src2addr.ab
                              01CA  ;       OPERAND_6(SP)
                              01CA  ;       OPERAND_7(SP)
                              01CA  ;       OPERAND_8(SP)
                              01CA  ;       NEW_PC(SP)
                              01CA  ;       EXCEPTION_PSL(SP)
                              01CA  ;
                              01CA  ; Output Parameters:
                              01CA  ;
                              01CA  ;       R0<15:0>    - srclen.rw
                              01CA  ;       R0<23:16>   - fill.rb
                              01CA  ;       R1          - srcaddr.ab
                              01CA  ;       R2<15:0>    - src2len.rw
                              01CA  ;       R3          - src2addr.ab
                              01CA  ;
                              01CA  ; Implicit Output:
                              01CA  ;
                              01CA  ;       R0<31:24> - UNPREDICTABLE
                              01CA  ;       R2<31:16> - 0
                              01CA  ;-
                              01CA
                              01CA  CMPC5:
                              01CA
         50   10 AE   10   9C 01CA          ROTL    #16,OPERAND_3(SP),R0    ; R0<23:16> <- fill.rb
              50   08 AE   B0 01CF          MOVW    OPERAND_1(SP),R0        ; R0<15:0>  <- src1len.rw
              51   0C AE   D0 01D3          MOVL    OPERAND_2(SP),R1        ; R1        <- src1addr.ab
              52   14 AE   3C 01D7          MOVZWL  OPERAND_4(SP),R2        ; R2<15:0>  <- src2len.rw
              53   18 AE   D0 01DB          MOVL    OPERAND_5(SP),R3        ; R3        <sca- src2addr.ab
                              01DF
                              01DF  ; Now that the operands have been loaded, the only exception parameter
                              01DF  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              01DF  ; there is no reason why the state of the stack needs to be altered and we
                              01DF  ; save two instructions if we leave the stack alone.
                              01DF
              0425'CF      9F 01DF          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
                 FE1A'     31 01E3          BRW     VAX$CMPC5               ; Do the actual work
```

```
                      01E6          .SUBTITLE       SCANC - Exception handler for SCANC instruction
                      01E6   ;+
                      01E6   ; Input Parameters:
                      01E6   ;
                      01E6   ;       OPCODE(SP)
                      01E6   ;       OLD_PC(SP)
                      01E6   ;       OPERAND_1(SP) - len.rw
                      01E6   ;       OPERAND_2(SP) - addr.ab
                      01E6   ;       OPERAND_3(SP) - tbladdr.ab
                      01E6   ;       OPERAND_4(SP) - mask.ab
                      01E6   ;       OPERAND_5(SP)
                      01E6   ;       OPERAND_6(SP)
                      01E6   ;       OPERAND_7(SP)
                      01E6   ;       OPERAND_8(SP)
                      01E6   ;       NEW_PC(SP)
                      01E6   ;       EXCEPTION_PSL(SP)
                      01E6   ;
                      01E6   ; Output Parameters:
                      01E6   ;
                      01E6   ;       R0<15:0> - len.rw
                      01E6   ;       R1       - addr.ab
                      01E6   ;       R2<7:0>  - mask.rb
                      01E6   ;       R3       - tbladdr.ab
                      01E6   ;
                      01E6   ; Implicit Output:
                      01E6   ;
                      01E6   ;       R0<31:16> - 0
                      01E6   ;       R2<31:8>  - 0
                      01E6   ;-
                      01E6
                      01E6   SCANC:
                      01E6
50   08 AE      3C    01E6          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- len.rw
51   0C AE      D0    01EA          MOVL    OPERAND_2(SP),R1        ; R1       <- addr.ab
53   10 AE      D0    01EE          MOVL    OPERAND_3(SP),R3        ; R3       <- tbladdr.ab
52   14 AE      9A    01F2          MOVZBL  OPERAND_4(SP),R2        ; R2<7:0>  <- mask.ab
                      01F6
                      01F6   ; Now that the operands have been loaded, the only exception parameter
                      01F6   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                      01F6   ; there is no reason why the state of the stack needs to be altered and we
                      01F6   ; save two instructions if we leave the stack alone.
                      01F6
   0425'CF      9F    01F6          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
      FE03'     31    01FA          BRW     VAX$SCANC               ; Do the actual work
```

```
                         01FD            .SUBTITLE      SPANC - Exception handler for SPANC instruction
                         01FD     ;+
                         01FD     ; Input Parameters:
                         01FD     ;
                         01FD     ;       OPCODE(SP)
                         01FD     ;       OLD_PC(SP)
                         01FD     ;       OPERAND_1(SP) - len.rw
                         01FD     ;       OPERAND_2(SP) - addr.ab
                         01FD     ;       OPERAND_3(SP) - tbladdr.ab
                         01FD     ;       OPERAND_4(SP) - mask.ab
                         01FD     ;       OPERAND_5(SP)
                         01FD     ;       OPERAND_6(SP)
                         01FD     ;       OPERAND_7(SP)
                         01FD     ;       OPERAND_8(SP)
                         01FD     ;       NEW_PC(SP)
                         01FD     ;       EXCEPTION_PSL(SP)
                         01FD     ;
                         01FD     ; Output Parameters:
                         01FD     ;
                         01FD     ;       R0<15:0> - len.rw
                         01FD     ;       R1       - addr.ab
                         01FD     ;       R2<7:0>  - mask.rb
                         01FD     ;       R3       - tbladdr.ab
                         01FD     ;
                         01FD     ; Implicit Output:
                         01FD     ;
                         01FD     ;       R0<31:16> - 0
                         01FD     ;       R2<31:8>  - 0
                         01FD     ;-
                         01FD
                         01FD     SPANC:
                         01FD
  50    08 AE    3C      01FD            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- len.rw
  51    0C AE    D0      0201            MOVL    OPERAND_2(SP),R1        ; R1       <- addr.ab
  53    10 AE    D0      0205            MOVL    OPERAND_3(SP),R3        ; R3       <- tbladdr.ab
  52    14 AE    9A      0209            MOVZBL  OPERAND_4(SP),R2        ; R2<7:0>  <- mask.ab
                         020D
                         020D     ; Now that the operands have been loaded, the only exception parameter
                         020D     ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         020D     ; there is no reason why the state of the stack needs to be altered and we
                         020D     ; save two instructions if we leave the stack alone.
                         020D
      0425'CF    9F      020D            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
         FDEC'   31      0211            BRW     VAX$SPANC               ; Do the actual work
```

```
                        0214          .SUBTITLE       LOCC - Exception handler for LOCC instruction
                        0214  ;+
                        0214  ; Input Parameters:
                        0214  ;
                        0214  ;       OPCODE(SP)
                        0214  ;       OLD_PC(SP)
                        0214  ;       OPERAND_1(SP) - char.rb
                        0214  ;       OPERAND_2(SP) - len.rw
                        0214  ;       OPERAND_3(SP) - addr.ab
                        0214  ;       OPERAND_4(SP)
                        0214  ;       OPERAND_5(SP)
                        0214  ;       OPERAND_6(SP)
                        0214  ;       OPERAND_7(SP)
                        0214  ;       OPERAND_8(SP)
                        0214  ;       NEW_PC(SP)
                        0214  ;       EXCEPTION_PSL(SP)
                        0214  ;
                        0214  ; Output Parameters:
                        0214  ;
                        0214  ;       R0<15:0>  - len.rw
                        0214  ;       R0<23:16> - char.rb
                        0214  ;       R1        - addr.ab
                        0214  ;
                        0214  ; Implicit Output:
                        0214  ;
                        0214  ;       R0<31:24> - UNPREDICTABLE
                        0214  ;-
                        0214
                        0214  LOCC:
                        0214
 50   08 AE   10   9C   0214          ROTL    #16,OPERAND_1(SP),R0      ; R0<23:16> <- char.ab
      50   0C AE   B0   0219          MOVW    OPERAND_2(SP),R0          ; R0<15:0>  <- len.rw
      51   10 AE   D0   021D          MOVL    OPERAND_3(SP),R1          ; R1        <- addr.ab
                        0221
                        0221  ; Now that the operands have been loaded, the only exception parameter
                        0221  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0221  ; there is no reason why the state of the stack needs to be altered and we
                        0221  ; save two instructions if we leave the stack alone.
                        0221
          0425'CF  9F   0221          PUSHAB  VAX$EXIT_EMULATOR         ; Store the return PC
            FDD8'  31   0225          BRW     VAX$LOCC                  ; Do the actual work
```

```
                               0228         .SUBTITLE       SKPC - Exception handler for SKPC instruction
                               0228   ;+
                               0228   ; Input Parameters:
                               0228   ;
                               0228   ;         OPCODE(SP)
                               0228   ;         OLD_PC(SP)
                               0228   ;         OPERAND_1(SP) - char.rb
                               0228   ;         OPERAND_2(SP) - len.rw
                               0228   ;         OPERAND_3(SP) - addr.ab
                               0228   ;         OPERAND_4(SP)
                               0228   ;         OPERAND_5(SP)
                               0228   ;         OPERAND_6(SP)
                               0228   ;         OPERAND_7(SP)
                               0228   ;         OPERAND_8(SP)
                               0228   ;         NEW_PC(SP)
                               0228   ;         EXCEPTION_PSL(SP)
                               0228   ;
                               0228   ; Output Parameters:
                               0228   ;
                               0228   ;         R0<15:0>  - len.rw
                               0228   ;         R0<23:16> - char.rb
                               0228   ;         R1        - addr.ab
                               0228   ;
                               0228   ; Implicit Output:
                               0228   ;
                               0228   ;         R0<31:24> - UNPREDICTABLE
                               0228   ;-
                               0228
                               0228   SKPC:
                               0228
 50    08 AE    10    9C       0228          ROTL    #16,OPERAND_1(SP),R0     ; R0<23:16> <- char.ab
       50    0C AE    B0       022D          MOVW    OPERAND_2(SP),R0         ; R0<15:0>  <- len.rw
       51    10 AE    D0       0231          MOVL    OPERAND_3(SP),R1         ; R1        <- addr.ab
                               0235
                               0235   ; Now that the operands have been loaded, the only exception parameter
                               0235   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                               0235   ; there is no reason why the state of the stack needs to be altered and we
                               0235   ; save two instructions if we leave the stack alone.
                               0235
          0425'CF     9F       0235          PUSHAB  VAX$EXIT_EMULATOR        ; Store the return PC
             FDC4'    31       0239          BRW     VAX$SKPC                 ; Do the actual work
```

```
                        023C            .SUBTITLE       MATCHC - Exception handler for MATCHC instruction
                        023C     ;+
                        023C     ; Input Parameters:
                        023C     ;
                        023C     ;      OPCODE(SP)
                        023C     ;      OLD_PC(SP)
                        023C     ;      OPERAND_1(SP) - objlen.rw
                        023C     ;      OPERAND_2(SP) - objaddr.ab
                        023C     ;      OPERAND_3(SP) - srclen.rw
                        023C     ;      OPERAND_4(SP) - srcaddr.ab
                        023C     ;      OPERAND_5(SP)
                        023C     ;      OPERAND_6(SP)
                        023C     ;      OPERAND_7(SP)
                        023C     ;      OPERAND_8(SP)
                        023C     ;      NEW_PC(SP)
                        023C     ;      EXCEPTION_PSL(SP)
                        023C     ;
                        023C     ; Output Parameters:
                        023C     ;
                        023C     ;      R0<15:0>  - objlen.rw
                        023C     ;      R1        - objaddr.ab
                        023C     ;      R2<15:0>  - srclen.rw
                        023C     ;      R3        - srcaddr.ab
                        023C     ;
                        023C     ; Implicit Output:
                        023C     ;
                        023C     ;      R0<31:16> - 0
                        023C     ;      R2<31:16> - 0
                        023C     ;-
                        023C
                        023C     MATCHC:
                        023C
   50    08 AE     3C   023C            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0>   <- objlen.rw
   51    0C AE     D0   0240            MOVL    OPERAND_2(SP),R1        ; R1         <- objaddr.ab
   52    10 AE     3C   0244            MOVZWL  OPERAND_3(SP),R2        ; R2<15:0>   <- srclen.rw
   53    14 AE     D0   0248            MOVL    OPERAND_4(SP),R3        ; R3         <- srcaddr.ab
                        024C
                        024C     ; Now that the operands have been loaded, the only exception parameter
                        024C     ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        024C     ; there is no reason why the state of the stack needs to be altered and we
                        024C     ; save two instructions if we leave the stack alone.
                        024C
       0425'CF     9F   024C            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
         FDAD'     31   0250            BRW     VAX$MATCHC              ; Do the actual work
```

```
                        0253         .SUBTITLE      CRC - Exception handler for CRC instruction
                        0253   ;+
                        0253   ; Input Parameters:
                        0253   ;
                        0253   ;        OPCODE(SP)
                        0253   ;        OLD_PC(SP)
                        0253   ;        OPERAND_1(SP) - tbl.ab
                        0253   ;        OPERAND_2(SP) - inicrc.rl
                        0253   ;        OPERAND_3(SP) - strlen.rw
                        0253   ;        OPERAND_4(SP) - stream.ab
                        0253   ;        OPERAND_5(SP)
                        0253   ;        OPERAND_6(SP)
                        0253   ;        OPERAND_7(SP)
                        0253   ;        OPERAND_8(SP)
                        0253   ;        NEW_PC(SP)
                        0253   ;        EXCEPTION_PSL(SP)
                        0253   ;
                        0253   ; Output Parameters:
                        0253   ;
                        0253   ;        R0          - inicrc.rl
                        0253   ;        R1          - tbl.ab
                        0253   ;        R2<15:0>    - strlen.rw
                        0253   ;        R3          - stream.ab
                        0253   ;
                        0253   ; Implicit Output:
                        0253   ;
                        0253   ;        R2<31:16> - 0
                        0253   ;-
                        0253
                        0253   CRC:
                        0253
  51    08 AE    D0     0253         MOVL     OPERAND_1(SP),R1        ; R1         <- tbl.ab
  50    0C AE    D0     0257         MOVL     OPERAND_2(SP),R0        ; R0         <- inicrc.rl
  52    10 AE    3C     025B         MOVZWL   OPERAND_3(SP),R2        ; R2<15:0>   <- strlen.rw
  53    14 AE    D0     025F         MOVL     OPERAND_4(SP),R3        ; R3         <- stream.ab
                        0263
                        0263   ; Now that the operands have been loaded, the only exception parameter
                        0263   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0263   ; there is no reason why the state of the stack needs to be altered and we
                        0263   ; save two instructions if we leave the stack alone.
                        0263
     0425'CF     9F     0263         PUSHAB   VAX$EXIT_EMULATOR       ; Store the return PC
        FD96'    31     0267         BRW      VAX$CRC                 ; Do the actual work
```

```
                               026A          .SUBTITLE       ADDP4 - Exception handler for ADDP4 instruction
                               026A    ;+
                               026A    ; Input Parameters:
                               026A    ;
                               026A    ;       OPCODE(SP)
                               026A    ;       OLD_PC(SP)
                               026A    ;       OPERAND_1(SP) - addlen.rw
                               026A    ;       OPERAND_2(SP) - addaddr.ab
                               026A    ;       OPERAND_3(SP) - sumlen.rw
                               026A    ;       OPERAND_4(SP) - sumaddr.ab
                               026A    ;       OPERAND_5(SP)
                               026A    ;       OPERAND_6(SP)
                               026A    ;       OPERAND_7(SP)
                               026A    ;       OPERAND_8(SP)
                               026A    ;       NEW_PC(SP)
                               026A    ;       EXCEPTION_PSL(SP)
                               026A    ;
                               026A    ; Output Parameters:
                               026A    ;
                               026A    ;       R0<15:0> - addlen.rw
                               026A    ;       R1       - addaddr.ab
                               026A    ;       R2<15:0> - sumlen.rw
                               026A    ;       R3       - sumaddr.ab
                               026A    ;
                               026A    ; Implicit Output:
                               026A    ;
                               026A    ;       R0<31:16> - 0
                               026A    ;       R2<31:16> - 0
                               026A    ;-
                               026A
                               026A   ADDP4:
                               026A
         50   08 AE     3C     026A          MOVZWL  OPERAND_1(SP),R0         ; R0<15:0> <- addlen.rw
         51   0C AE     D0     026E          MOVL    OPERAND_2(SP),R1         ; R1       <- addaddr.ab
         52   10 AE     3C     0272          MOVZWL  OPERAND_3(SP),R2         ; R2<15:0> <- sumlen.rw
         53   14 AE     D0     0276          MOVL    OPERAND_4(SP),R3         ; R3       <- sumaddr.ab
                               027A
                               027A    ; Now that the operands have been loaded, the only exception parameter
                               027A    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                               027A    ; there is no reason why the state of the stack needs to be altered and we
                               027A    ; save two instructions if we leave the stack alone.
                               027A
             0425'CF    9F     027A          PUSHAB  VAX$EXIT_EMULATOR        ; Store the return PC
               FD7F'    31     027E          BRW     VAX$ADDP4                ; Do the actual work
```

```
                          0281          .SUBTITLE       ADDP6 - Exception handler for ADDP6 instruction
                          0281  ;+
                          0281  ; Input Parameters:
                          0281  ;
                          0281  ;       OPCODE(SP)
                          0281  ;       OLD_PC(SP)
                          0281  ;       OPERAND_1(SP) - add1len.rw
                          0281  ;       OPERAND_2(SP) - add1addr.ab
                          0281  ;       OPERAND_3(SP) - add2len.rw
                          0281  ;       OPERAND_4(SP) - add2addr.ab
                          0281  ;       OPERAND_5(SP) - sumlen.rw
                          0281  ;       OPERAND_6(SP) - sumaddr.ab
                          0281  ;       OPERAND_7(SP)
                          0281  ;       OPERAND_8(SP)
                          0281  ;       NEW_PC(SP)
                          0281  ;       EXCEPTION_PSL(SP)
                          0281  ;
                          0281  ; Output Parameters:
                          0281  ;
                          0281  ;       R0<15:0> - add1len.rw
                          0281  ;       R1       - add1addr.ab
                          0281  ;       R2<15:0> - add2len.rw
                          0281  ;       R3       - add2addr.ab
                          0281  ;       R4<15:0> - sumlen.rw
                          0281  ;       R5       - sumaddr.ab
                          0281  ;
                          0281  ; Implicit Output:
                          0281  ;
                          0281  ;       R0<31:16> - 0
                          0281  ;       R2<31:16> - 0
                          0281  ;       R4<31:16> - 0
                          0281  ;-
                          0281
                          0281  ADDP6:
                          0281
50     08 AE      3C      0281          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- add1len.rw
51     0C AE      D0      0285          MOVL    OPERAND_2(SP),R1        ; R1       <- add1addr.ab
52     10 AE      3C      0289          MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- add2len.rw
53     14 AE      D0      028D          MOVL    OPERAND_4(SP),R3        ; R3       <- add2addr.ab
54     18 AE      3C      0291          MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- sumlen.rw
55     1C AE      D0      0295          MOVL    OPERAND_6(SP),R5        ; R5       <- sumaddr.ab
                          0299
                          0299  ; Now that the operands have been loaded, the only exception parameter
                          0299  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          0299  ; there is no reason why the state of the stack needs to be altered and we
                          0299  ; save two instructions if we leave the stack alone.
                          0299
      0425'CF     9F      0299          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
        FD60'     31      029D          BRW     VAX$ADDP6               ; Do the actual work
```

```
                                 02A0          .SUBTITLE       ASHP - Exception handler for ASHP instruction
                                 02A0  ;+
                                 02A0  ; Input Parameters:
                                 02A0  ;
                                 02A0  ;       OPCODE(SP)
                                 02A0  ;       OLD_PC(SP)
                                 02A0  ;       OPERAND_1(SP) - cnt.rb
                                 02A0  ;       OPERAND_2(SP) - srclen.rw
                                 02A0  ;       OPERAND_3(SP) - srcaddr.ab
                                 02A0  ;       OPERAND_4(SP) - round.rb
                                 02A0  ;       OPERAND_5(SP) - dstlen.rw
                                 02A0  ;       OPERAND_6(SP) - dstaddr.ab
                                 02A0  ;       OPERAND_7(SP)
                                 02A0  ;       OPERAND_8(SP)
                                 02A0  ;       NEW_PC(SP)
                                 02A0  ;       EXCEPTION_PSL(SP)
                                 02A0  ;
                                 02A0  ; Output Parameters:
                                 02A0  ;
                                 02A0  ;       R0<15:0>  - srclen.rw
                                 02A0  ;       R0<31:16> - count.rb
                                 02A0  ;       R1        - srcaddr.ab
                                 02A0  ;       R2<15:0>  - dstlen.rw
                                 02A0  ;       R2<31:16> - round.rb
                                 02A0  ;       R3        - dstaddr.ab
                                 02A0  ;
                                 02A0  ; Implicit Output:
                                 02A0  ;
                                 02A0  ;       R0<31:24> - UNPREDICTABLE
                                 02A0  ;       R2<31:24> - UNPREDICTABLE
                                 02A0  ;-
                                 02A0
                                 02A0  ASHP:
                                 02A0
      50   08 AE    10    9C     02A0          ROTL    #16,OPERAND_1(SP),R0    ; R0<31:16> <- count.rb
           50 0C AE       B0     02A5          MOVW    OPERAND_2(SP),R0        ; R0<15:0>  <- srclen.rw
           51 10 AE       D0     02A9          MOVL    OPERAND_3(SP),R1        ; R1        <- srcaddr.ab
      52   14 AE    10    9C     02AD          ROTL    #16,OPERAND_4(SP),R2    ; R2<31:16> <- round.rb
           52 18 AE       B0     02B2          MOVW    OPERAND_5(SP),R2        ; R2<15:0>  <- dstlen.rw
           53 1C AE       D0     02B6          MOVL    OPERAND_6(SP),R3        ; R3        <- dstaddr.ab
                                 02BA
                                 02BA  ; Now that the operands have been loaded, the only exception parameter
                                 02BA  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                 02BA  ; there is no reason why the state of the stack needs to be altered and we
                                 02BA  ; save two instructions if we leave the stack alone.
                                 02BA
             0425'CF      9F     02BA          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
                FD3F'     31     02BE          BRW     VAX$ASHP                ; Do the actual work
```

```
                         02C1          .SUBTITLE       CMPP3 - Exception handler for CMPP3 instruction
                         02C1    ;+
                         02C1    ; Input Parameters:
                         02C1    ;
                         02C1    ;       OPCODE(SP)
                         02C1    ;       OLD_PC(SP)
                         02C1    ;       OPERAND_1(SP) - len.rw
                         02C1    ;       OPERAND_2(SP) - src1addr.ab
                         02C1    ;       OPERAND_3(SP) - src2addr.ab
                         02C1    ;       OPERAND_4(SP)
                         02C1    ;       OPERAND_5(SP)
                         02C1    ;       OPERAND_6(SP)
                         02C1    ;       OPERAND_7(SP)
                         02C1    ;       OPERAND_8(SP)
                         02C1    ;       NEW_PC(SP)
                         02C1    ;       EXCEPTION_PSL(SP)
                         02C1    ;
                         02C1    ; Output Parameters:
                         02C1    ;
                         02C1    ;       R0<15:0> - len.rw
                         02C1    ;       R1       - src1addr.ab
                         02C1    ;       R3       - src2addr.ab
                         02C1    ;
                         02C1    ; Implicit Output:
                         02C1    ;
                         02C1    ;       R0<31:16> - 0
                         02C1    ;       R2        - UNPREDICTABLE
                         02C1    ;-
                         02C1
                         02C1    CMPP3:
                         02C1
     50   08 AE    3C    02C1          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- len.rw
     51   0C AE    D0    02C5          MOVL    OPERAND_2(SP),R1        ; R1       <- src1addr.ab
     53   10 AE    D0    02C9          MOVL    OPERAND_3(SP),R3        ; R3       <- src2addr.ab
                         02CD
                         02CD    ; Now that the operands have been loaded, the only exception parameter
                         02CD    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         02CD    ; there is no reason why the state of the stack needs to be altered and we
                         02CD    ; save two instructions if we leave the stack alone.
                         02CD
       0425'CF     9F    02CD          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
          FD2C'    31    02D1          BRW     VAX$CMPP3               ; Do the actual work
```

```
                    02D4          .SUBTITLE       CMPP4 - Exception handler for CMPP4 instruction
                    02D4   ;+
                    02D4   ; Input Parameters:
                    02D4   ;
                    02D4   ;          OPCODE(SP)
                    02D4   ;          OLD_PC(SP)
                    02D4   ;          OPERAND_1(SP) - src1len.rw
                    02D4   ;          OPERAND_2(SP) - src1addr.ab
                    02D4   ;          OPERAND_3(SP) - src2len.rw
                    02D4   ;          OPERAND_4(SP) - src2addr.ab
                    02D4   ;          OPERAND_5(SP)
                    02D4   ;          OPERAND_6(SP)
                    02D4   ;          OPERAND_7(SP)
                    02D4   ;          OPERAND_8(SP)
                    02D4   ;          NEW_PC(SP)
                    02D4   ;          EXCEPTION_PSL(SP)
                    02D4   ;
                    02D4   ; Output Parameters:
                    02D4   ;
                    02D4   ;          R0<15:0> - src1len.rw
                    02D4   ;          R1       - src1addr.ab
                    02D4   ;          R2<15:0> - src2len.rw
                    02D4   ;          R3       - src2addr.ab
                    02D4   ;
                    02D4   ; Implicit Output:
                    02D4   ;
                    02D4   ;          R0<31:16> - 0
                    02D4   ;          R2<31:16> - 0
                    02D4   ;-
                    02D4
                    02D4   CMPP4:
                    02D4
  50   08 AE   3C   02D4          MOVZWL  OPERAND_1(SP),R0         ; R0<15:0> <- src1len.rw
  51   0C AE   D0   02D8          MOVL    OPERAND_2(SP),R1         ; R1       <- src1addr.ab
  52   10 AE   3C   02DC          MOVZWL  OPERAND_3(SP),R2         ; R2<15:0> <- src2len.rw
  53   14 AE   D0   02E0          MOVL    OPERAND_4(SP),R3         ; R3       <- src2addr.ab
                    02E4
                    02E4   ; Now that the operands have been loaded, the only exception parameter
                    02E4   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                    02E4   ; there is no reason why the state of the stack needs to be altered and we
                    02E4   ; save two instructions if we leave the stack alone.
                    02E4
     0425'CF   9F   02E4          PUSHAB  VAX$EXIT_EMULATOR        ; Store the return PC
       FD15'   31   02E8          BRW     VAX$CMPP4                ; Do the actual work
```

```
                          02EB          .SUBTITLE       CVTLP - Exception handler for CVTLP instruction
                          02EB  ;+
                          02EB  ; Input Parameters:
                          02EB  ;
                          02EB  ;       OPCODE(SP)
                          02EB  ;       OLD_PC(SP)
                          02EB  ;       OPERAND_1(SP) - src.rl
                          02EB  ;       OPERAND_2(SP) - dstlen.rw
                          02EB  ;       OPERAND_3(SP) - dstaddr.ab
                          02EB  ;       OPERAND_4(SP)
                          02EB  ;       OPERAND_5(SP)
                          02EB  ;       OPERAND_6(SP)
                          02EB  ;       OPERAND_7(SP)
                          02EB  ;       OPERAND_8(SP)
                          02EB  ;       NEW_PC(SP)
                          02EB  ;       EXCEPTION_PSL(SP)
                          02EB  ;
                          02EB  ; Output Parameters:
                          02EB  ;
                          02EB  ;       R0          - src.rl
                          02EB  ;       R2<15:0> - dstlen.rw
                          02EB  ;       R3          - dstaddr.ab
                          02EB  ;
                          02EB  ; Implicit Output:
                          02EB  ;
                          02EB  ;       R1          - explicitly set to zero
                          02EB  ;       R2<31:16> - 0
                          02EB  ;-
                          02EB
                          02EB  CVTLP:
                          02EB
50    08 AE     D0        02EB          MOVL    OPERAND_1(SP),R0        ; R0        <- src.rl
           51   D4        02EF          CLRL    R1                      ; R1        <- 0
52    0C AE     3C        02F1          MOVZWL  OPERAND_2(SP),R2        ; R2<15:0> <- dstlen.rw
53    10 AE     D0        02F5          MOVL    OPERAND_3(SP),R3        ; R3        <- dstaddr.ab
                          02F9
                          02F9  ; Now that the operands have been loaded, the only exception parameter
                          02F9  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                          02F9  ; there is no reason why the state of the stack needs to be altered and we
                          02F9  ; save two instructions if we leave the stack alone.
                          02F9
   0425'CF      9F        02F9          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
      FD00'     31        02FD          BRW     VAX$CVTLP               ; Do the actual work
```

```
                        0300            .SUBTITLE       CVTPL - Exception handler for CVTPL instruction
                        0300    ;+
                        0300    ; Input Parameters:
                        0300    ;
                        0300    ;       OPCODE(SP)
                        0300    ;       OLD_PC(SP)
                        0300    ;       OPERAND_1(SP) - srclen.rw
                        0300    ;       OPERAND_2(SP) - srcaddr.ab
                        0300    ;       OPERAND_3(SP) - dst.wl
                        0300    ;       OPERAND_4(SP)
                        0300    ;       OPERAND_5(SP)
                        0300    ;       OPERAND_6(SP)
                        0300    ;       OPERAND_7(SP)
                        0300    ;       OPERAND_8(SP)
                        0300    ;       NEW_PC(SP)
                        0300    ;       EXCEPTION_PSL(SP)
                        0300    ;
                        0300    ; Output Parameters:
                        0300    ;
                        0300    ;       R0<15:0> - srclen.rw
                        0300    ;       R1       - srcaddr.ab
                        0300    ;       R3       - dst.wl
                        0300    ;
                        0300    ; Notes:
                        0300    ;
                        0300    ;       The routine header for VAX$CVTPL describes how the destination is
                        0300    ;       encoded in a register. Basically, OPERAND_3 contains the effective
                        0300    ;       address of the operand. If the destination is a general register, then
                        0300    ;       OPERAND_3 contains the ones complement of the register number.
                        0300    ;
                        0300    ; Implicit Output:
                        0300    ;
                        0300    ;       R0<31:16> - 0
                        0300    ;       R2        - explicitly set to zero
                        0300    ;-
                        0300
                        0300    CVTPL:
                        0300
 50    08 AE    3C      0300            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
 51    0C AE    D0      0304            MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
          52    D4      0308            CLRL    R2                      ; R2       <- 0
 53    10 AE    D0      030A            MOVL    OPERAND_3(SP),R3        ; R3       <- dst.wl
                        030E
                        030E    ; Now that the operands have been loaded, the only exception parameter
                        030E    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        030E    ; there is no reason why the state of the stack needs to be altered and we
                        030E    ; save two instructions if we leave the stack alone.
                        030E
     0425'CF    9F      030E            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
       FCEB'    31      0312            BRW     VAX$CVTPL               ; Do the actual work
```

```
                     0315        .SUBTITLE       CVTPS - Exception handler for CVTPS instruction
                     0315  ;+
                     0315  ; Input Parameters:
                     0315  ;
                     0315  ;       OPCODE(SP)
                     0315  ;       OLD_PC(SP)
                     0315  ;       OPERAND_1(SP) - srclen.rw
                     0315  ;       OPERAND_2(SP) - srcaddr.ab
                     0315  ;       OPERAND_3(SP) - dstlen.rw
                     0315  ;       OPERAND_4(SP) - dstaddr.ab
                     0315  ;       OPERAND_5(SP)
                     0315  ;       OPERAND_6(SP)
                     0315  ;       OPERAND_7(SP)
                     0315  ;       OPERAND_8(SP)
                     0315  ;       NEW_PC(SP)
                     0315  ;       EXCEPTION_PSL(SP)
                     0315  ;
                     0315  ; Output Parameters:
                     0315  ;
                     0315  ;       R0<15:0> - srclen.rw
                     0315  ;       R1       - srcaddr.ab
                     0315  ;       R2<15:0> - dstlen.rw
                     0315  ;       R3       - dstaddr.ab
                     0315  ;
                     0315  ; Implicit Output:
                     0315  ;
                     0315  ;       R0<31:16> - 0
                     0315  ;       R2<31:16> - 0
                     0315  ;-
                     0315
                     0315  CVTPS:
                     0315
50    08 AE    3C    0315        MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
51    0C AE    D0    0319        MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
52    10 AE    3C    031D        MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- dstlen.rw
53    14 AE    D0    0321        MOVL    OPERAND_4(SP),R3        ; R3       <- dstaddr.ab
                     0325
                     0325  ; Now that the operands have been loaded, the only exception parameter
                     0325  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                     0325  ; there is no reason why the state of the stack needs to be altered and we
                     0325  ; save two instructions if we leave the stack alone.
                     0325
   0425'CF     9F    0325        PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
      FCD4'    31    0329        BRW     VAX$CVTPS               ; Do the actual work
```

```
                              032C          .SUBTITLE       CVTPT - Exception handler for CVTPT instruction
                              032C    ;+
                              032C    ; Input Parameters:
                              032C    ;
                              032C    ;       OPCODE(SP)
                              032C    ;       OLD_PC(SP)
                              032C    ;       OPERAND_1(SP) - srclen.rw
                              032C    ;       OPERAND_2(SP) - srcaddr.ab
                              032C    ;       OPERAND_3(SP) - tbladdr.ab
                              032C    ;       OPERAND_4(SP) - dstlen.rw
                              032C    ;       OPERAND_5(SP) - dstaddr.ab
                              032C    ;       OPERAND_6(SP)
                              032C    ;       OPERAND_7(SP)
                              032C    ;       OPERAND_8(SP)
                              032C    ;       NEW_PC(SP)
                              032C    ;       EXCEPTION_PSL(SP)
                              032C    ;
                              032C    ; Output Parameters:
                              032C    ;
                              032C    ;       R0<15:0>  - srclen.rw
                              032C    ;       R0<31:16> - dstlen.rw
                              032C    ;       R1        - srcaddr.ab
                              032C    ;       R2        - tbladdr.ab
                              032C    ;       R3        - dstaddr.ab
                              032C    ;-
                              032C
                              032C    CVTPT:
                              032C
   50   14 AE   10    9C      032C          ROTL    #16,OPERAND_4(SP),R0    ; R0<31:16> <- dstlen.rw
        50  08 AE     B0      0331          MOVW    OPERAND_1(SP),R0        ; R0<15:0>  <- srclen.rw
        51  0C AE     D0      0335          MOVL    OPERAND_2(SP),R1        ; R1        <- srcaddr.ab
        52  10 AE     D0      0339          MOVL    OPERAND_3(SP),R2        ; R2        <- tbladdr.ab
        53  18 AE     D0      033D          MOVL    OPERAND_5(SP),R3        ; R3        <- dstaddr.ab
                              0341
                              0341    ; Now that the operands have been loaded, the only exception parameter
                              0341    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              0341    ; there is no reason why the state of the stack needs to be altered and we
                              0341    ; save two instructions if we leave the stack alone.
                              0341
           0425'CF    9F      0341          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
              FCB8'   31      0345          BRW     VAX$CVTPT               ; Do the actual work
```

```
                            0348         .SUBTITLE       CVTSP - Exception handler for CVTSP instruction
                            0348  ;+
                            0348  ; Input Parameters:
                            0348  ;
                            0348  ;        OPCODE(SP)
                            0348  ;        OLD_PC(SP)
                            0348  ;        OPERAND_1(SP) - srclen.rw
                            0348  ;        OPERAND_2(SP) - srcaddr.ab
                            0348  ;        OPERAND_3(SP) - dstlen.rw
                            0348  ;        OPERAND_4(SP) - dstaddr.ab
                            0348  ;        OPERAND_5(SP)
                            0348  ;        OPERAND_6(SP)
                            0348  ;        OPERAND_7(SP)
                            0348  ;        OPERAND_8(SP)
                            0348  ;        NEW_PC(SP)
                            0348  ;        EXCEPTION_PSL(SP)
                            0348  ;
                            0348  ; Output Parameters:
                            0348  ;
                            0348  ;        R0<15:0> - srclen.rw
                            0348  ;        R1       - srcaddr.ab
                            0348  ;        R2<15:0> - dstlen.rw
                            0348  ;        R3       - dstaddr.ab
                            0348  ;
                            0348  ; Implicit Output:
                            0348  ;
                            0348  ;        R0<31:16> - 0
                            0348  ;        R2<31:16> - 0
                            0348  ;-
                            0348
                            0348  CVTSP:
                            0348
  50    08 AE    3C         0348         MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
  51    0C AE    D0         034C         MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
  52    10 AE    3C         0350         MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- dstlen.rw
  53    14 AE    D0         0354         MOVL    OPERAND_4(SP),R3        ; R3       <- dstaddr.ab
                            0358
                            0358  ; Now that the operands have been loaded, the only exception parameter
                            0358  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                            0358  ; there is no reason why the state of the stack needs to be altered and we
                            0358  ; save two instructions if we leave the stack alone.
                            0358
     0425'CF     9F         0358         PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
       FCA1'     31         035C         BRW     VAX$CVTSP               ; Do the actual work
```

```
                              035F          .SUBTITLE       CVTTP - Exception handler for CVTTP instruction
                              035F  ;+
                              035F  ; Input Parameters:
                              035F  ;
                              035F  ;       OPCODE(SP)
                              035F  ;       OLD_PC(SP)
                              035F  ;       OPERAND_1(SP) - srclen.rw
                              035F  ;       OPERAND_2(SP) - srcaddr.ab
                              035F  ;       OPERAND_3(SP) - tbladdr.ab
                              035F  ;       OPERAND_4(SP) - dstlen.rw
                              035F  ;       OPERAND_5(SP) - dstaddr.ab
                              035F  ;       OPERAND_6(SP)
                              035F  ;       OPERAND_7(SP)
                              035F  ;       OPERAND_8(SP)
                              035F  ;       NEW_PC(SP)
                              035F  ;       EXCEPTION_PSL(SP)
                              035F  ;
                              035F  ; Output Parameters:
                              035F  ;
                              035F  ;       R0<15:0>  - srclen.rw
                              035F  ;       R0<31:16> - dstlen.rw
                              035F  ;       R1        - srcaddr.ab
                              035F  ;       R2        - tbladdr.ab
                              035F  ;       R3        - dstaddr.ab
                              035F  ;-
                              035F
                              035F  CVTTP:
                              035F
       50    14 AE    10   9C 035F          ROTL    #16,OPERAND_4(SP),R0     ; R0<31:16> <- dstlen.rw
             50 08 AE      B0 0364          MOVW    OPERAND_1(SP),R0         ; R0<15:0>  <- srclen.rw
             51 0C AE      D0 0368          MOVL    OPERAND_2(SP),R1         ; R1        <- srcaddr.ab
             52 10 AE      D0 036C          MOVL    OPERAND_3(SP),R2         ; R2        <- tbladdr.ab
             53 18 AE      D0 0370          MOVL    OPERAND_5(SP),R3         ; R3        <- dstaddr.ab
                              0374
                              0374  ; Now that the operands have been loaded, the only exception parameter
                              0374  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                              0374  ; there is no reason why the state of the stack needs to be altered and we
                              0374  ; save two instructions if we leave the stack alone.
                              0374
             0425'CF       9F 0374          PUSHAB  VAX$EXIT_EMULATOR        ; Store the return PC
                FC85'      31 0378          BRW     VAX$CVTTP                ; Do the actual work
```

```
                        037B            .SUBTITLE       DIVP - Exception handler for DIVP instruction
                        037B    ;+
                        037B    ; Input Parameters:
                        037B    ;
                        037B    ;       OPCODE(SP)
                        037B    ;       OLD_PC(SP)
                        037B    ;       OPERAND_1(SP) - divrlen.rw
                        037B    ;       OPERAND_2(SP) - divraddr.ab
                        037B    ;       OPERAND_3(SP) - divdlen.rw
                        037B    ;       OPERAND_4(SP) - divdaddr.ab
                        037B    ;       OPERAND_5(SP) - quolen.rw
                        037B    ;       OPERAND_6(SP) - quoaddr.ab
                        037B    ;       OPERAND_7(SP)
                        037B    ;       OPERAND_8(SP)
                        037B    ;       NEW_PC(SP)
                        037B    ;       EXCEPTION_PSL(SP)
                        037B    ;
                        037B    ; Output Parameters:
                        037B    ;
                        037B    ;       R0<15:0> - divrlen.rw
                        037B    ;       R1       - divraddr.ab
                        037B    ;       R2<15:0> - divdlen.rw
                        037B    ;       R3       - divdaddr.ab
                        037B    ;       R4<15:0> - quolen.rw
                        037B    ;       R5       - quoaddr.ab
                        037B    ;
                        037B    ; Implicit Output:
                        037B    ;
                        037B    ;       R0<31:16> - 0
                        037B    ;       R2<31:16> - 0
                        037B    ;       R4<31:16> - 0
                        037B    ;-
                        037B
                        037B    DIVP:
                        037B
  50    08 AE    3C     037B            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- divrlen.rw
  51    0C AE    D0     037F            MOVL    OPERAND_2(SP),R1        ; R1       <- divraddr.ab
  52    10 AE    3C     0383            MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- divdlen.rw
  53    14 AE    D0     0387            MOVL    OPERAND_4(SP),R3        ; R3       <- divdaddr.ab
  54    18 AE    3C     038B            MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- quolen.rw
  55    1C AE    D0     038F            MOVL    OPERAND_6(SP),R5        ; R5       <- quoaddr.ab
                        0393
                        0393    ; Now that the operands have been loaded, the only exception parameter
                        0393    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        0393    ; there is no reason why the state of the stack needs to be altered and we
                        0393    ; save two instructions if we leave the stack alone.
                        0393
     0425'CF     9F     0393            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
        FC66'    31     0397            BRW     VAX$DIVP                ; Do the actual work
```

```
                        039A          .SUBTITLE       MOVP - Exception handler for MOVP instruction
                        039A  ;+
                        039A  ; Input Parameters:
                        039A  ;
                        039A  ;       OPCODE(SP)
                        039A  ;       OLD_PC(SP)
                        039A  ;       OPERAND_1(SP) - len.rw
                        039A  ;       OPERAND_2(SP) - srcaddr.ab
                        039A  ;       OPERAND_3(SP) - dstaddr.ab
                        039A  ;       OPERAND_4(SP)
                        039A  ;       OPERAND_5(SP)
                        039A  ;       OPERAND_6(SP)
                        039A  ;       OPERAND_7(SP)
                        039A  ;       OPERAND_8(SP)
                        039A  ;       NEW_PC(SP)
                        039A  ;       EXCEPTION_PSL(SP)
                        039A  ;
                        039A  ; Output Parameters:
                        039A  ;
                        039A  ;       R0<15:0> - len.rw
                        039A  ;       R1       - srcaddr.ab
                        039A  ;       R3       - dstaddr.ab
                        039A  ;
                        039A  ; Implicit Output:
                        039A  ;
                        039A  ;       R0<31:16> - 0
                        039A  ;       R2        - UNPREDICTABLE
                        039A  ;-
                        039A
                        039A  MOVP:
                        039A
    50   08 AE    3C    039A          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- len.rw
    51   0C AE    D0    039E          MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
    53   10 AE    D0    03A2          MOVL    OPERAND_3(SP),R3        ; R3       <- dstaddr.ab
                        03A6
                        03A6  ; Now that the operands have been loaded, the only exception parameter
                        03A6  ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        03A6  ; there is no reason why the state of the stack needs to be altered and we
                        03A6  ; save two instructions if we leave the stack alone.
                        03A6
                        03A6  ; The MOVP instruction is the only instruction in this entire set that
                        03A6  ; preserves the setting of the C-bit. The C-bit setting in the saved PSL
                        03A6  ; is propogated into the current PSL because the current PSL forms the
                        03A6  ; initial setting for the final settings of the condition codes.
                        03A6
            01    B9    03A6          BICPSW  #PSL$M_C                        ; Assume C bit is clear
                        03A8
                        03A8          ASSUME PSL$V_C EQ 0                     ; Make sure that BLBC is OK
                        03A8
      02 2C AE    E9    03A8          BLBC    EXCEPTION_PSL(SP),10$   ; Skip next if saved C-bit is clear
            01    B8    03AC          BISPSW  #PSL$M_C                        ; Otherwise, set the C-bit
                        03AE
                        03AE  ; Note that it is crucial that no instructions that alter the C-bit can
                        03AE  ; execute until the PSL is saved in VAX$MOVP. PUSHAB preserves the C-bit.
                        03AE
       0425'CF    9F    03AE  10$:    PUSHAB  VAX$EXIT_EMULATOR               ; Store the return PC
         FC4B'    31    03B2          BRW     VAX$MOVP                        ; Do the actual work
```

```
                           03B5          .SUBTITLE      MULP - Exception handler for MULP instruction
                           03B5   ;+
                           03B5   ; Input Parameters:
                           03B5   ;
                           03B5   ;      OPCODE(SP)
                           03B5   ;      OLD_PC(SP)
                           03B5   ;      OPERAND_1(SP) - mulrlen.rw
                           03B5   ;      OPERAND_2(SP) - mulraddr.ab
                           03B5   ;      OPERAND_3(SP) - muldlen.rw
                           03B5   ;      OPERAND_4(SP) - muldaddr.ab
                           03B5   ;      OPERAND_5(SP) - prodlen.rw
                           03B5   ;      OPERAND_6(SP) - prodaddr.ab
                           03B5   ;      OPERAND_7(SP)
                           03B5   ;      OPERAND_8(SP)
                           03B5   ;      NEW_PC(SP)
                           03B5   ;      EXCEPTION_PSL(SP)
                           03B5   ;
                           03B5   ; Output Parameters:
                           03B5   ;
                           03B5   ;      R0<15:0> - mulrlen.rw
                           03B5   ;      R1       - mulraddr.ab
                           03B5   ;      R2<15:0> - muldlen.rw
                           03B5   ;      R3       - muldaddr.ab
                           03B5   ;      R4<15:0> - prodlen.rw
                           03B5   ;      R5       - prodaddr.ab
                           03B5   ;
                           03B5   ; Implicit Output:
                           03B5   ;
                           03B5   ;      R0<31:16> - 0
                           03B5   ;      R2<31:16> - 0
                           03B5   ;      R4<31:16> - 0
                           03B5   ;-
                           03B5
                           03B5  MULP:
                           03B5
       50   08 AE   3C     03B5          MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- mulrlen.rw
       51   0C AE   D0     03B9          MOVL    OPERAND_2(SP),R1        ; R1       <- mulraddr.ab
       52   10 AE   3C     03BD          MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- muldlen.rw
       53   14 AE   D0     03C1          MOVL    OPERAND_4(SP),R3        ; R3       <- muldaddr.ab
       54   18 AE   3C     03C5          MOVZWL  OPERAND_5(SP),R4        ; R4<15:0> <- prodlen.rw
       55   1C AE   D0     03C9          MOVL    OPERAND_6(SP),R5        ; R5       <- prodaddr.ab
                           03CD
                           03CD   ; Now that the operands have been loaded, the only exception parameter
                           03CD   ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                           03CD   ; there is no reason why the state of the stack needs to be altered and we
                           03CD   ; save two instructions if we leave the stack alone.
                           03CD
          0425'CF   9F     03CD          PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
             FC2C'  31     03D1          BRW     VAX$MULP                ; Do the actual work
```

```
                        03D4          .SUBTITLE       SUBP4 - Exception handler for SUBP4 instruction
                        03D4    ;+
                        03D4    ; Input Parameters:
                        03D4    ;
                        03D4    ;       OPCODE(SP)
                        03D4    ;       OLD_PC(SP)
                        03D4    ;       OPERAND_1(SP) - sublen.rw
                        03D4    ;       OPERAND_2(SP) - subaddr.ab
                        03D4    ;       OPERAND_3(SP) - diflen.rw
                        03D4    ;       OPERAND_4(SP) - difaddr.ab
                        03D4    ;       OPERAND_5(SP)
                        03D4    ;       OPERAND_6(SP)
                        03D4    ;       OPERAND_7(SP)
                        03D4    ;       OPERAND_8(SP)
                        03D4    ;       NEW_PC(SP)
                        03D4    ;       EXCEPTION_PSL(SP)
                        03D4    ;
                        03D4    ; Output Parameters:
                        03D4    ;
                        03D4    ;       R0<15:0> - sublen.rw
                        03D4    ;       R1       - subaddr.ab
                        03D4    ;       R2<15:0> - diflen.rw
                        03D4    ;       R3       - difaddr.ab
                        03D4    ;
                        03D4    ; Implicit Output:
                        03D4    ;
                        03D4    ;       R0<31:16> - 0
                        03D4    ;       R2<31:16> - 0
                        03D4    ;-
                        03D4
                        03D4    SUBP4:
                        03D4
   50    08 AE    3C    03D4            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- sublen.rw
   51    0C AE    D0    03D8            MOVL    OPERAND_2(SP),R1        ; R1       <- subaddr.ab
   52    10 AE    3C    03DC            MOVZWL  OPERAND_3(SP),R2        ; R2<15:0> <- diflen.rw
   53    14 AE    D0    03E0            MOVL    OPERAND_4(SP),R3        ; R3       <- difaddr.ab
                        03E4
                        03E4    ; Now that the operands have been loaded, the only exception parameter
                        03E4    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                        03E4    ; there is no reason why the state of the stack needs to be altered and we
                        03E4    ; save two instructions if we leave the stack alone.
                        03E4
      0425'CF     9F    03E4            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
         FC15'    31    03E8            BRW     VAX$SUBP4               ; Do the actual work
```

```
                         03EB          .SUBTITLE       SUBP6 - Exception handler for SUBP6 instruction
                         03EB    ;+
                         03EB    ; Input Parameters:
                         03EB    ;
                         03EB    ;       OPCODE(SP)
                         03EB    ;       OLD_PC(SP)
                         03EB    ;       OPERAND_1(SP) - sublen.rw
                         03EB    ;       OPERAND_2(SP) - subaddr.ab
                         03EB    ;       OPERAND_3(SP) - minlen.rw
                         03EB    ;       OPERAND_4(SP) - minaddr.ab
                         03EB    ;       OPERAND_5(SP) - diflen.rw
                         03EB    ;       OPERAND_6(SP) - difaddr.ab
                         03EB    ;       OPERAND_7(SP)
                         03EB    ;       OPERAND_8(SP)
                         03EB    ;       NEW_PC(SP)
                         03EB    ;       EXCEPTION_PSL(SP)
                         03EB    ;
                         03EB    ; Output Parameters:
                         03EB    ;
                         03EB    ;       R0<15:0> - sublen.rw
                         03EB    ;       R1       - subaddr.ab
                         03EB    ;       R2<15:0> - minlen.rw
                         03EB    ;       R3       - minaddr.ab
                         03EB    ;       R4<15:0> - diflen.rw
                         03EB    ;       R5       - difaddr.ab
                         03EB    ;
                         03EB    ; Implicit Output:
                         03EB    ;
                         03EB    ;       R0<31:16> - 0
                         03EB    ;       R2<31:16> - 0
                         03EB    ;       R4<31:16> - 0
                         03EB    ;-
                         03EB
                         03EB    SUBP6:
                         03EB
  50    08 AE    3C      03EB           MOVZWL  OPERAND_1(SP),R0         ; R0<15:0> <- sublen.rw
  51    0C AE    D0      03EF           MOVL    OPERAND_2(SP),R1         ; R1       <- subaddr.ab
  52    10 AE    3C      03F3           MOVZWL  OPERAND_3(SP),R2         ; R2<15:0> <- minlen.rw
  53    14 AE    D0      03F7           MOVL    OPERAND_4(SP),R3         ; R3       <- minaddr.ab
  54    18 AE    3C      03FB           MOVZWL  OPERAND_5(SP),R4         ; R4<15:0> <- diflen.rw
  55    1C AE    D0      03FF           MOVL    OPERAND_6(SP),R5         ; R5       <- difaddr.ab
                         0403
                         0403    ; Now that the operands have been loaded, the only exception parameter
                         0403    ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                         0403    ; there is no reason why the state of the stack needs to be altered and we
                         0403    ; save two instructions if we leave the stack alone.
                         0403
     0425'CF     9F      0403           PUSHAB  VAX$EXIT_EMULATOR        ; Store the return PC
        FBF6'    31      0407           BRW     VAX$SUBP6                ; Do the actual work
```

```
                                040A            .SUBTITLE       EDITPC - Exception handler for EDITPC instruction
                                040A      ;+
                                040A      ; Input Parameters:
                                040A      ;
                                040A      ;       OPCODE(SP)
                                040A      ;       OLD_PC(SP)
                                040A      ;       OPERAND_1(SP) - srclen.rw
                                040A      ;       OPERAND_2(SP) - srcaddr.ab
                                040A      ;       OPERAND_3(SP) - pattern.ab
                                040A      ;       OPERAND_4(SP) - dstaddr.ab
                                040A      ;       OPERAND_5(SP)
                                040A      ;       OPERAND_6(SP)
                                040A      ;       OPERAND_7(SP)
                                040A      ;       OPERAND_8(SP)
                                040A      ;       NEW_PC(SP)
                                040A      ;       EXCEPTION_PSL(SP)
                                040A      ;
                                040A      ; Output Parameters:
                                040A      ;
                                040A      ;       R0<15:0> - srclen.rw
                                040A      ;       R1       - srcaddr.ab
                                040A      ;       R3       - pattern.ab
                                040A      ;       R5       - dstaddr.ab
                                040A      ;
                                040A      ; Implicit Output:
                                040A      ;
                                040A      ;       R0<31:16> - 0
                                040A      ;       R2        - explicitly set to zero
                                040A      ;       R4        - explicitly set to zero
                                040A      ;-
                                040A
                                040A      EDITPC:
                                040A
      50    08 AE      3C       040A            MOVZWL  OPERAND_1(SP),R0        ; R0<15:0> <- srclen.rw
      51    0C AE      D0       040E            MOVL    OPERAND_2(SP),R1        ; R1       <- srcaddr.ab
               52      D4       0412            CLRL    R2                      ; R2       <- 0
      53    10 AE      D0       0414            MOVL    OPERAND_3(SP),R3        ; R3       <- pattern.ab
               54      D4       0418            CLRL    R4                      ; R4       <- 0
      55    14 AE      D0       041A            MOVL    OPERAND_4(SP),R5        ; R5       <- dstaddr.ab
                                041E
                                041E      ; Now that the operands have been loaded, the only exception parameter
                                041E      ; other than the PC/PSL pair that needs to be saved is the old PC. However,
                                041E      ; there is no reason why the state of the stack needs to be altered and we
                                041E      ; save two instructions if we leave the stack alone.
                                041E
         0425'CF       9F       041E            PUSHAB  VAX$EXIT_EMULATOR       ; Store the return PC
            FBDB'      31       0422            BRW     VAX$EDITPC              ; Do the actual work
```

```
                          0425          .SUBTITLE       Common Exit Path for VAX$xxxxxx Routines
                          0425  ;+
                          0425  ; Functional Description:
                          0425  ;
                          0425  ;       This is the common exit path for all instruction-specific routines.
                          0425  ;       The condition codes returned by the VAX$xxxxxx routine are stored in
                          0425  ;       the exception PSL and control is passed back to the instruction stream
                          0425  ;       that executed the reserved instruction.
                          0425  ;
                          0425  ; Input Parameters:
                          0425  ;
                          0425  ;       PSL contains condition code settings from VAX$xxxxxx routine.
                          0425  ;
                          0425  ;       OPCODE(SP)      - Opcode of reserved instruction
                          0425  ;       OLD_PC(SP)      - PC of reserved instruction
                          0425  ;       OPERAND_1(SP) - First operand specifier
                          0425  ;       OPERAND_2(SP) - Second operand specifier
                          0425  ;       OPERAND_3(SP) - Third operand specifier
                          0425  ;       OPERAND_4(SP) - Fourth operand specifier
                          0425  ;       OPERAND_5(SP) - Fifth operand specifier
                          0425  ;       OPERAND_6(SP) - Sixth operand specifier
                          0425  ;       OPERAND_7(SP) - Seventh operand specifier (currently unused)
                          0425  ;       OPERAND_8(SP) - Eight operand specifier (currently unused)
                          0425  ;       NEW_PC(SP)      - PC of instruction following reserved instruction
                          0425  ;       EXCEPTION_PSL(SP) - PSL at time of exception
                          0425  ;
                          0425  ; Implicit Input:
                          0425  ;
                          0425  ;       General registers contain architecturally specified values according
                          0425  ;       to specific instruction that was emulated.
                          0425  ;
                          0425  ; Implicit Output:
                          0425  ;
                          0425  ;       Control is passed to the location designated by "new PC" with the
                          0425  ;       condition codes as determined by VAX$xxxxxx. The EXIT routine also
                          0425  ;       preserves general registers.
                          0425  ;-
                          0425
                          0425  VAX$EXIT_EMULATOR::
                 7E   DC  0425          MOVPSL  -(SP)                   ; Save the new PSL on the stack
                          0427
                          0427  ; Note that the next instruction makes no assumptions about the condition
                          0427  ; codes in the saved PSL.
                          0427
     04   00   8E   F0  0427          INSV    (SP)+,#0,#4,-
            2C   AE       0428                   EXCEPTION_PSL(SP)       ; Replace saved condition codes
           5E   28   C0   042D          ADDL    #NEW_PC,SP              ; Adjust stack pointer (discard old PC)
                     02   0430          REI                             ; Return
                          0431
                          0431          .END
```

```
                                                        L 14
VAX$EMULATE                         - VAX-11 Instruction Emulator          16-SEP-1984 01:29:10  VAX/VMS Macro V04-00        Page  42
Symbol table                                                                5-SEP-1984 00:45:28  [EMULAT.SRC]VAXEMULAT.MAR;1       (34)

...OFFSET                       = 00000040                 OP$_ADDP4                   = 00000020
...OPCODE                       = 00000038                 OP$_ADDP6                   = 00000021
ADDP4                             0000026A R      02       OP$_ASHP                    = 000000F8
ADDP6                             00000281 R      02       OP$_CMPC3                   = 00000029
ASHP                              000002A0 R      02       OP$_CMPC5                   = 0000002D
CASE_TABLE_BASE                   00000006 R      02       OP$_CMPP3                   = 00000035
CASE_TABLE_SIZE                 = 00000044                 OP$_CMPP4                   = 00000037
CMPC3                             000001B7 R      02       OP$_CRC                     = 0000000B
CMPC5                             000001CA R      02       OP$_CVTLP                   = 000000F9
CMPP3                             000002C1 R      02       OP$_CVTPL                   = 00000036
CMPP4                             000002D4 R      02       OP$_CVTPS                   = 00000008
CRC                               00000253 R      02       OP$_CVTPT                   = 00000024
CVTLP                             000002EB R      02       OP$_CVTSP                   = 00000009
CVTPL                             00000300 R      02       OP$_CVTTP                   = 00000026
CVTPS                             00000315 R      02       OP$_DIVP                    = 00000027
CVTPT                             0000032C R      02       OP$_EDITPC                  = 00000038
CVTSP                             00000348 R      02       OP$_LOCC                    = 0000003A
CVTTP                             0000035F R      02       OP$_MATCHC                  = 00000039
DIVP                              0000037B R      02       OP$_MOVP                    = 00000034
EDITPC                            0000040A R      02       OP$_MOVTC                   = 0000002E
EXCEPTION_PSL                   = 0000002C                 OP$_MOVTUC                  = 0000002F
FPD_CASE_TABLE_BASE               000000D6 R      02       OP$_MULP                    = 00000025
INCLUDE_ADDP4                   = 00000000                 OP$_SCANC                   = 0000002A
INCLUDE_ADDP6                   = 00000000                 OP$_SKPC                    = 0000003B
INCLUDE_ASHP                    = 00000000                 OP$_SPANC                   = 0000002B
INCLUDE_CMPC3                   = 00000000                 OP$_SUBP4                   = 00000022
INCLUDE_CMPC5                   = 00000000                 OP$_SUBP6                   = 00000023
INCLUDE_CMPP3                   = 00000000                 OPCODE                      = 00000000
INCLUDE_CMPP4                   = 00000000                 OPCODE_BASE                 = FFFFFFF8
INCLUDE_CRC                     = 00000000                 OPCODE_MAX                  = 0000003B
INCLUDE_CVTLP                   = 00000000                 OPERAND_1                   = 00000008
INCLUDE_CVTPL                   = 00000000                 OPERAND_2                   = 0000000C
INCLUDE_CVTPS                   = 00000000                 OPERAND_3                   = 00000010
INCLUDE_CVTPT                   = 00000000                 OPERAND_4                   = 00000014
INCLUDE_CVTSP                   = 00000000                 OPERAND_5                   = 00000018
INCLUDE_CVTTP                   = 00000000                 OPERAND_6                   = 0000001C
INCLUDE_DIVP                    = 00000000                 OPERAND_8                   = 00000024
INCLUDE_EDITPC                  = 00000000                 PSL$M_C                     = 00000001
INCLUDE_LOCC                    = 00000000                 PSL$V_C                     = 00000000
INCLUDE_MATCHC                  = 00000000                 PSL$V_FPD                   = 0000001B
INCLUDE_MOVP                    = 00000000                 SCANC                         000001E6 R      02
INCLUDE_MOVTC                   = 00000000                 SKPC                          00000228 R      02
INCLUDE_MOVTUC                  = 00000000                 SPANC                         000001FD R      02
INCLUDE_MULP                    = 00000000                 SUBP4                         000003D4 R      02
INCLUDE_SCANC                   = 00000000                 SUBP6                         000003EB R      02
INCLUDE_SKPC                    = 00000000                 VAX$ADDP4                     ********    X   02
INCLUDE_SPANC                   = 00000000                 VAX$ADDP6                     ********    X   02
INCLUDE_SUBP4                   = 00000000                 VAX$AL_DELTA_PC_TABLE         ********    X   00
INCLUDE_SUBP6                   = 00000000                 VAX$ASHP                      ********    X   02
LOCC                              00000214 R      02       VAX$CMPC3                     ********    X   02
MATCHC                            0000023C R      02       VAX$CMPC5                     ********    X   02
MOVP                              0000039A R      02       VAX$CMPP3                     ********    X   02
MOVTC                             00000179 R      02       VAX$CMPP4                     ********    X   02
MOVTUC                            00000198 R      02       VAX$CRC                       ********    X   02
MULP                              000003B5 R      02       VAX$CVTLP                     ********    X   02
NEW_PC                          = 00000028                 VAX$CVTLP_RESTART             ********    X   02
OLD_PC                          = 00000004                 VAX$CVTPL                     ********    X   02
```

```
                                                            M 14
VAX$EMULATE                 - VAX-11 Instruction Emulator          16-SEP-1984 01:29:10  VAX/VMS Macro V04-00        Page  43
Symbol table                                                        5-SEP-1984 00:45:28  [EMULAT.SRC]VAXEMULAT.MAR;1      (34)

VAX$CVTPL_RESTART                ********   X   02
VAX$CVTPS                        ********   X   02
VAX$CVTPT                        ********   X   02
VAX$CVTPT_RESTART                ********   X   02
VAX$CVTSP                        ********   X   02
VAX$CVTTP                        ********   X   02
VAX$CVTTP_RESTART                ********   X   02
VAX$DIVP                         ********   X   02
VAX$EDITPC                       ********   X   02
VAX$EDITPC_RESTART               ********   X   02
VAX$EMULATE                      00000000 RG    02
VAX$EMULATE_FPD                  0000009C RG    02
VAX$EXIT_EMULATOR                00000425 RG    02
VAX$LOCC                         ********   X   02
VAX$MATCHC                       ********   X   02
VAX$MOVP                         ********   X   02
VAX$MOVTC                        ********   X   02
VAX$MOVTUC                       ********   X   02
VAX$MULP                         ********   X   02
VAX$REFLECT_TO_VMS               ********   X   00
VAX$SCANC                        ********   X   02
VAX$SKPC                         ********   X   02
VAX$SPANC                        ********   X   02
VAX$SUBP4                        ********   X   02
VAX$SUBP6                        ********   X   02
VAX$_OPCDEC                      ********   X   00
VAX$_OPCDEC_FPD                  ********   X   00

                                              +----------------+
                                              ! Psect synopsis !
                                              +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_VAX$CODE                       00000431  ( 1073.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC QUAD

                                           +------------------------+
                                           ! Performance indicators !
                                           +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                     9    00:00:00.05    00:00:00.76
Command processing                84    00:00:00.57    00:00:05.27
Pass 1                           271    00:00:09.05    00:00:29.52
Symbol table sort                  0    00:00:00.71    00:00:03.06
Pass 2                           316    00:00:04.15    00:00:12.58
Symbol table output               16    00:00:00.12    00:00:00.39
Psect synopsis output              2    00:00:00.02    00:00:00.02
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals             698    00:00:14.67    00:00:51.60

The working set limit was 1500 pages.
66591 bytes (131 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 532 non-local and 4 local symbols.
```

```
2002 source lines were read in Pass 1, producing 21 object records in Pass 2.
16 pages of virtual memory were used to define 14 macros.

                                          +----------------------------+
                                          ! Macro library statistics !
                                          +----------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[EMULAT.OBJ]VAXMACROS.MLB;1               3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     6
TOTALS (all libraries)                                9

524 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:VAXEMULAT/OBJ=OBJ$:VAXEMULAT MSRC$:VAXEMULAT/UPDATE=(ENH$:VAXEMULAT)+LIB$:VAXMACROS/LIB
```

0144 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

VAXHANDLR
LIS

VAXCVTLP
LIS

VAXDECIML
LIS

VAXEMULAT
LIS

VAXCVTPL
LIS

VAXEDITPC
LIS